NASA Technical Memorandum 100687

A Computer Code to Process and Plot Laser Altimetry Data Interactively on a Microcomputer

H. G. Safren and J. L. Bufton
Goddard Space Flight Center
Greenbelt, Maryland

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

1987

CONTENTS

A COMPUTER CODE TO PROCESS AND PLOT LASER ALTIMETRY DATA INTERACTIVELY ON A MICROCOMPUTER

H. G. Safren and J. L. Bufton

INTRODUCTION

The computer program described in this report was developed to process and plot data taken with a laser altimeter currently under development in the Instrument Electro-Optics Branch of the Goddard Space Flight Center.

The altimeter is being tested by flying it in an aircraft at about 30,000 feet over mountainous terrain. The raw data consist of pulse timing information and round trip return times; these must be converted into along-track position information and ground height above sea level. At the present stage of development there is no position or attitude information for the aircraft, so the data must be processed on the assumption that the aircraft is flying a straight path at a constant altitude and that it maintains a horizontal attitude while data are being taken.

The computer program consists of three main routines, plus two auxiliary routines and various command files to execute the routines in the proper sequence. The first main routine unpacks the raw data, which are stored two numbers to a byte. The second routine adjusts the data point times, which are given only to the preceding integer second at present, by using the fact that the time between laser pulses is constant, and then converts the pulse times and the return times into along-track distance and ground height. The third routine plots the processed data to show ground height versus along-track distance; in other words, it shows the terrain profile along the aircraft's path. The auxiliary routines allow the user to inspect raw data files and to create a list of such files for use by the main routines. Each raw data file consists of four hundred and fifty data points; each data point consists of the time of day and the return time of a laser pulse.

The computer program is implemented for a specific microcomputer system—a Digital Equipment Corporation 11/23, using DEC's RT11 operating system, with dual, 8-inch floppy disk drives, a 10 megabyte RL02 hard disk and a VT100 terminal retrofitted with a graphics enhancement board manufactured by the Digital Engineering Corporation (Sacramento, CA). The program uses a plot package developed specifically for this computer system by the author, so that the program cannot be adapted to a different computer without replacing the plot routine. The program is highly interactive, and uses many features of the microcomputer to achieve a high degree of user control over the plot displays.

The three main routines and the two auxiliary routines reside on a floppy disk, called the program disk. Data sets are also placed on floppy disks. The two command files are placed in a different location (a hard disk), because they direct the mounting of the program and data disks and thus must always be accessible.

The plot routine has a kind of zoom capability; the user may select any part of the data and replot only that part. Thus the plot may in effect be expanded to show any section in greater detail. In addition, the vertical axis scale may be changed until the plot has the appearance desired by the user. Thus, the terrain profile may be shown in one-to-one scale to give a real picture of its appearance, or it may be exaggerated vertically to any degree to clearly show height variations. Provision is also made to allow the user to interactively edit the data set by deleting any data points which appear to be spurious; this eliminates the necessity of constructing complex algorithms for editing out undesired data points. Besides simplifying the code, the interactive editing probably is more effective, because the user can look at the plotted points and easily see which points are spurious.

DESCRIPTION OF PROGRAM

Since the program consists of three independent routines and two independent auxiliary routines, with command files to execute them in the proper sequence, the simplest way to describe the program is to describe each component separately.

Preprocessing Routine

This routine, called UNPACK, was adapted from an existing routine which had been written to unpack the raw data, which are stored two integer numbers to a byte. Each raw data file contains 450 data points; associated with each data point is certain other system information which we have no occasion to use here.

The raw data files are stored on floppy disks, in the order in which they were taken during the flight. The first disk contains a special file which sequentially lists all the raw data files for that flight, along with the sequence numbers of the disks on which they reside. The UNPACK routine queries the user for the sequence numbers of the first and last files to be processed; any contiguous subset of the data may be chosen. These files are then treated as a single set of data and are processed together; UNPACK automatically reads and unpacks the chosen files, in sequence, from the proper disks, instructing the user to mount disks when necessary. The entire set of unpacked data from all the chosen files is placed into a single file, for later use by the processing routine PROCES. This file is placed in a standard location (the Digital Equipment Corporation RL02, a hard disk drive) and is given a standard name, because the processing routine PROCES is coded to operate on a standard file.

Processing Routine

This routine, called PROCES, operates on the file of upacked data created by the preprocessing routine UNPACK. PROCES begins by asking the user to specify the values of several system parameters which are needed to process the data: the altitude above mean sea level of the ground point at which the first data point is taken, which is needed to compute ground heights along the flight path, the aircraft speed and the value of the effective index of refraction over the vertical path of the laser beam, which may be taken to be very nearly unity. These parameter values are then written as a header to the file which is to contain the processed data.

PROCES then reads the unpacked data in groups of 450 data points, processes them and writes the processed points (each of which consists of the along-track distance and the height above mean sea level of the ground point) to the file of processed data.

The processing is done in several steps. The first step is to adjust the times at which the data points are taken. This is necessary because each time is given only to the preceding (integer) second, and about seven data points are taken each second, so that several data points will generally have the same time in seconds associated with them. To improve these times, PROCES finds the first and last data points (in the group of 450 points) for which the value of seconds (the time is given in integer hours, minutes and seconds) changes; we will call these points point a and point b, for convenience. Assuming temporarily that the integer second values associated with points a and b are exact, better estimates are calculated for the times for the first and last points in the group (of 450 points) by assuming that seven data points are taken per second, which is a good approximation. For example, if point a happens to be the third point in the group, then the improved time for the first point would be calculated by subtracting 2/7ths of a second from the time for point a. The time between data points, which we will call delta, is then computed by dividing the difference of the corrected times of the last and first points by 449. Using this value of delta, improved time values for all of the 450 points are computed by assuming that the time originally associated with point a is exact, and adding or subtracting the appropriate number of deltas to get the time for each of the other points.

The next step in the processing is to convert the data times and the pulse round trip times to along-track distance in kilometers and height of the ground point above mean sea level in meters. These conversions are straightforward, and are clearly explained by comments in the program listing.

In the raw data files, there are some points which were clearly bad; these are marked in the raw data by assigning a zero to the pulse return time. These points are used by PROCES in the processing, because dropping them would destroy the sequence of data point times, but they are not written to the file of processed data. The processed data file is stored in a standard location (the Digital Equipment Corporation RL02 hard disk unit) and is given a standard name, because the plot routine PLOT is coded to operate on a standard file.

Plot Routine

This routine, called PLOT, operates either with a file of processed data created by PROCES or with a file of processed and possibly edited (by removing spurious data points) data created by PLOT itself in a previous plotting/editing session. In either case, the file is stored in a standard location (the RL02 hard disk unit) and is given a standard name, because PLOT is coded to operate on a standard file.

PLOT begins by reading the header of the processed file, which contains the values of the system parameters

used in creating that file, and displays them on the screen for the user. PLOT then proceeds to create an initial plot.

The initial plot shows the entire set of data in the processed file on one plot. If the flight path was a long one, the terrain profile shown on this plot will be very compressed along the horizontal axis. This initial plot allows the user to scan the terrain profile and choose which sections of it he wishes to examine in more detail.

At this point the user is presented with a sequence of three options; any, all or none of them may be chosen. The options are, in order:

1. Remove any points deemed to be spurious, by using a movable crosshair to identify the points to be removed;
2. Zoom in on any desired section of the plot by specifying a subrange (of the along-path distance) to be plotted; the sub-range may be specified either by typing the values of the end-points or by pointing to them with the movable crosshair;
3. Choose a vertical scale. There are three sub-options for choosing the scale:
   a. Use a standard scale, which extends from 100 meters below the lowest ground point (in the subrange to be plotted) or sea level, whichever is less, to 1,000 meters above the highest ground point (this option must be chosen if the user wants to display all of the outlying, spurious points, because otherwise they might be automatically clipped and not displayed);
   b. Define a new vertical scale, by typing the lowest and highest elevations (below or above sea level) to be displayed;
   c. Keep the present vertical scale.

If none of the three options are chosen by the user, the current plot is held on the screen until the user types a "G" (for "go") at the keyboard. If one or more of them is chosen, PLOT automatically erases the current plot and replots the data (either all of it or whatever subrange might have been specified by the user), in accordance with the user's instructions. For example, the new plot will not show any spurious points that were removed, and it will plot only the chosen subrange of data with the chosen vertical scale. At this point the user will again be presented with the above three options; this cycle will continue until the user chooses none of them. PLOT remembers the options that were last used; thus if an option is not chosen, the next replot will use the most recently specified values. For example, the user may repeatedly replot with different vertical scales; the subrange will remain the same until he changes it. Also, a larger "subrange," as well as a smaller one, may be specified.

If none of the three options is chosen, and the user types a G, PLOT asks the user if he wishes to continue examining this data file. If the reply is yes, PLOT erases the screen, then again displays the initial plot and the whole process starts all over again. If the reply is no, PLOT asks the user if he wishes to save the processed (and possibly edited) file. If the reply to this question is no, the file is discarded. If the reply is yes, the user is asked what name he wishes the file to be given and where it is to be stored, and is given a chance to mount a floppy disk, if necessary. PLOT then copies the processed/edited file to the specified location, along with the header with the system parameter values, but minus any spurious points that were deleted during the session.

PLOT then asks the user if he wishes to examine another processed/edited file. If the reply is no, the program terminates. If it is yes, PLOT calls the system subroutine SETCMD, which executes a system monitor command, stored in an array in PLOT itself, to run the command file RPLOT.COM (see below) which runs PLOT again. Thus the user may examine, successively, any number of processed/edited data files; if desired, a data file may be repeatedly edited, each time deleting more spurious points from the data set.

Auxiliary Routine to Inspect Raw Data Files

This routine, called SEERAW, is a modified version of UNPACK; it is designed to be run by the command file SEERAW.COM. These routines, and the auxiliary FORTRAN routine SEEAUX, must all be described together, because they form a loop which allows the user to successively inspect any number of raw data files, in any order.

When SEERAW.COM is executed, it first instructs the user to mount the disk containing the raw data file to be inspected and the program disk (which contains the three main routines and the two auxiliary routines) in the appropriate drives. It then displays the directory of the raw data disk, so the user can find the file name of the data file to be inspected. It next runs the SEERAW routine, which queries the user for the name of the file to be inspected. SEERAW then opens the file, reads the 450 data points from it, unpacks them and stores the unpacked data in a temporary file.

Control then passes back to SEERAW.COM, which calls the screen-oriented system editor to allow the user to inspect the unpacked file; the user may scroll the file backward or forward to examine the data. When the user exits from the editor, control again passes back to SEERAW.COM, which then deletes the temporary unpacked file and runs the auxiliary FORTRAN routine SEEAUX. This routine asks the user if he wants to examine another raw data file. If the reply is no, the program terminates. If it is yes, SEEAUX calls the system subroutine SETCMD, which executes a monitor command, stored in an array in SEEAUX, which executes SEERAW.COM again. Thus, the user may loop through the examination of any sequence of raw data files, which may be stored on different disks.

Auxiliary Routine to Create List of Raw Data Files

This routine, called CRELST (CREate LiST), allows the user to create a file, on the first raw data disk for a given flight, which lists the series of raw data files for that flight, along with the sequence number of the disk on which each resides. This file, which is just a directory of the raw files for the given flight, is used by the routine UNPACK; when the user gives UNPACK a set of data to be unpacked by specifying the sequence numbers of the first and last files, UNPACK uses the directory file to find the raw data files it needs. During execution, the directory file is copied to a scratch disk (actually a logical disk on a DEC RL02 hard disk unit), because the first raw data disk may be dismounted during the execution of UNPACK.

CRELST is run by the command file CRELST.COM. When this command file is run, it first tells the user to mount the first raw data disk for the given flight on the appropriate drive. It then runs CRELST, which leads the user through the process of entering the raw data file names and the sequence numbers of the disks on which they reside into the directory file. After the user has typed in the last file and told CRELST, in reply to its query, that there are no more files, control passes back to CRELST.COM, which then displays the disk directory of the first raw data disk. This directory will now contain the directory file just created.

Command File to Process and Plot Raw Data

This command file, called RALTIM.COM, directs the entire process of unpacking raw data, processing it and plotting it. When RALTIM.COM is executed, it first displays instructions to the user, directing him to make sure he has the raw data files to be processed on hand, to mount the required floppy disks in the appropriate drives, etc.

RALTIM.COM then displays the disk directory of the first raw data disk, so that the user has the names of the raw data files at hand to help in deciding which group of files is to be processed. (Note that this is the disk directory placed on the disk by the operating system, not the directory of raw data files created by CRELST.) RALTIM.COM then copies the directory of files created by CRELST (for use by UNPACK) to a scratch disk, in case the first raw data disk is later dismounted.

RALTIM.COM then successively executes the three routines—UNPACK, PROCES and PLOT.

Command File to Plot Previously Processed Data

This command file, called RPLOT.COM, directs the plotting of a previously processed data file. Upon execution, RPLOT.COM first directs the user to mount the program disk in the appropriate drive. It then calls the system editor to allow the user to place the name of the data file in the auxiliary command file COPY.COM; this command file displays instructions to the user to mount the disk containing that data file in the appropriate drive. After the user has placed the name of the file in COPY.COM and exited from the editor, control reverts to RPLOT.COM, which then actually executes COPY.COM (which now contains the name of the data file). COPY.COM, upon being actually executed, copies the data file to the RL02 unit (where it is assumed to be by PLOT) and gives it a standard name, which is also assumed by PLOT. RPLOT.COM then executes PLOT, which operates on the data file.

PROGRAM SIZE AND OPERATING SPEED

The three main routines (the preprocessing, processing and plotting routines) require about 36.4, 27.2 and 34.9 kilobytes of memory. However, because the three routines are executed consecutively and only one of them is in memory at any time, only 36.4 kilobytes of memory are required to run the program.

The auxiliary routines to inspect raw data files and to create a list of raw data files require about 35.4 and 10.3 kilobytes of memory. Only one of these routines will be in memory at any time (with none of the main routines present), so that the maximum memory required is still 36.4 kilobytes.

The time required to process data and produce a plot depends, of course, on the size of the data set. The plot shown in Figure 1 shows a terrain profile over a horizontal range of 140 kilometers, constructed from about 5,000 data points (i.e., laser shots); to preprocess, process and plot this data set required about 14 minutes of computer time.

HOW TO USE THE PROGRAM

The first step in using the program is to place the raw data files for a given flight on floppy disk(s), in the order in which the data files were originally taken. The files may be given any names. The series of files may contain data sets separated by large time intervals because of problems encountered during the flight, and may even contain initial calibration files. These will be obvious when the data are inspected. Large time gaps will clearly appear when the plot routine produces its initial plot. When the user has examined the data, a subset of "good" data may be identified for later use.

To examine the raw data files without processing them, the user needs only to execute the command file SEERAW.COM, which together with the routines it calls, leads the user through the process of inspecting the

LASER ALTIMETER PROFILE DATA

TRANSECT OF MOUNTAIN RANGES IN VIRGINIA: OCTOBER 7, 1985

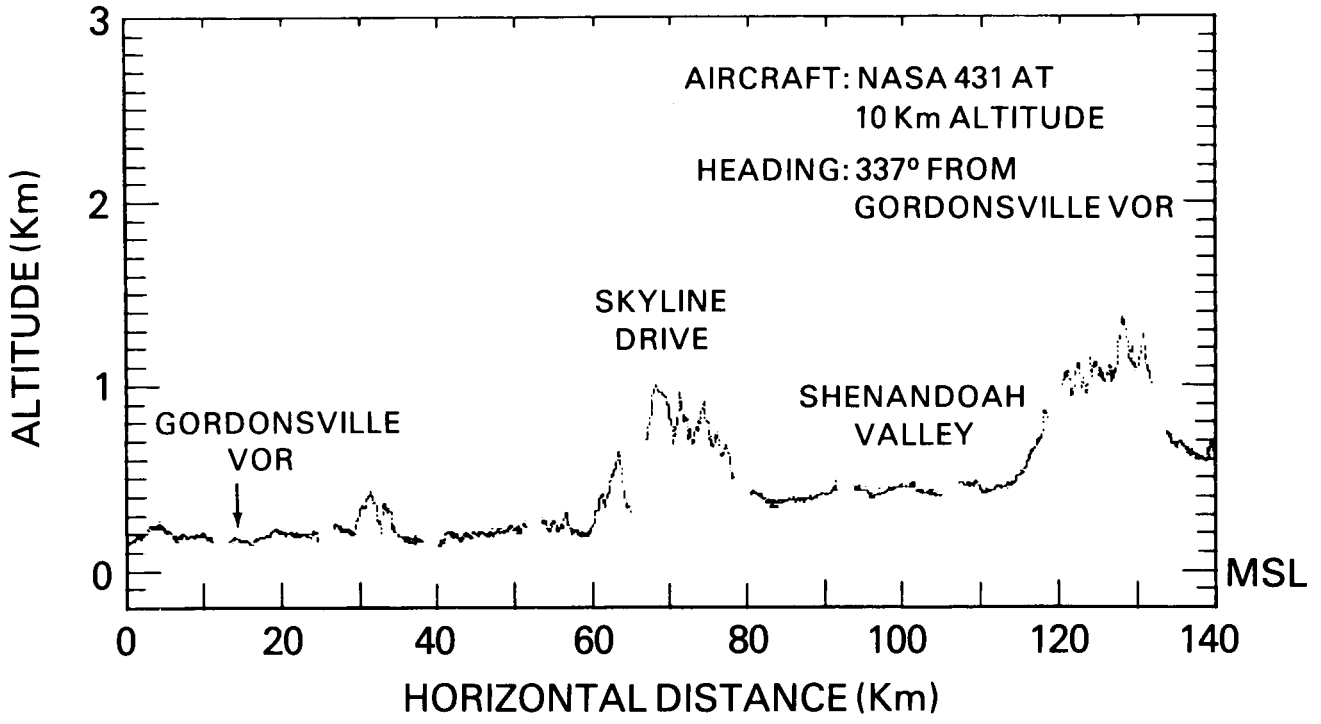

Figure 1. Initial Plot—Profile of Mountain Ranges in Virginia: October 7, 1985

raw data files.

Given a disk or a series of disks containing the series of raw data files for a given flight, the next step is to use the routine CRELST to create a directory of files on the first disk, for later use by the unpacking routine UNPACK. CRELST is run by executing its command file, CRELST.COM which, together with CRELST itself, leads the user step-by-step through the process of creating the directory of files.

The raw data are ready to be processed at this point. The user needs only to execute the command file RALTIM.COM which, together with the routines it calls, leads the user through the process of preprocessing (unpacking), processing and plotting the data.

If the user wishes to plot (and possibly edit) a file of already processed data, it is only necessary to execute the command file RPLOT.COM, which together with the routines it calls leads the user through the process of setting up the existing processed data file and plotting and, if desired, editing it.

Throughout the execution of the program, especially in the plot routine, the user will encounter places where nothing seems to happen; for example, a plot will just remain on the screen. To proceed to the next step in such a case, the user should type G and then hit the RETURN key. This method of operation was coded into the FORTRAN routines by using the ACCEPT statement, for the purpose of allowing the user to decide when to proceed.

SOME EXAMPLES OF TERRAIN PLOTS PRODUCED BY THE PROGRAM

Figure 1 shows the initial plot produced by the program (with some labeling that was added later) for a flight over the mountains of Virginia on October 7, 1985. The plot shows the entire data set. There are eleven data records in the data set, each consisting of 450 laser pulses. The separation between data records is due to a time interval between records of approximately 11 seconds. Note that the vertical scale is greatly exaggerated relative to the horizontal scale, in order to clearly show the vertical structure.

Figures 2 through 4 illustrate the capability of the plot routine to show successively smaller subranges of the

data. Figure 2 shows the Skyline Drive region, which was covered by one data record (450 laser pulses, with a total duration of 65 seconds). The altitude is relative to mean sea level (MSL). Figure 3 shows a subrange of Figure 2, and Figure 4 shows a subrange of Figure 3. Note that in Figure 4 the laser pulses are clearly separated on the plot.

Figure 5 illustrates the capability of the plot routine to vary the vertical scale; in this figure the vertical scale was adjusted to very nearly match the horizontal scale, to show a realistic view of the terrain profile in the Skyline Drive region.

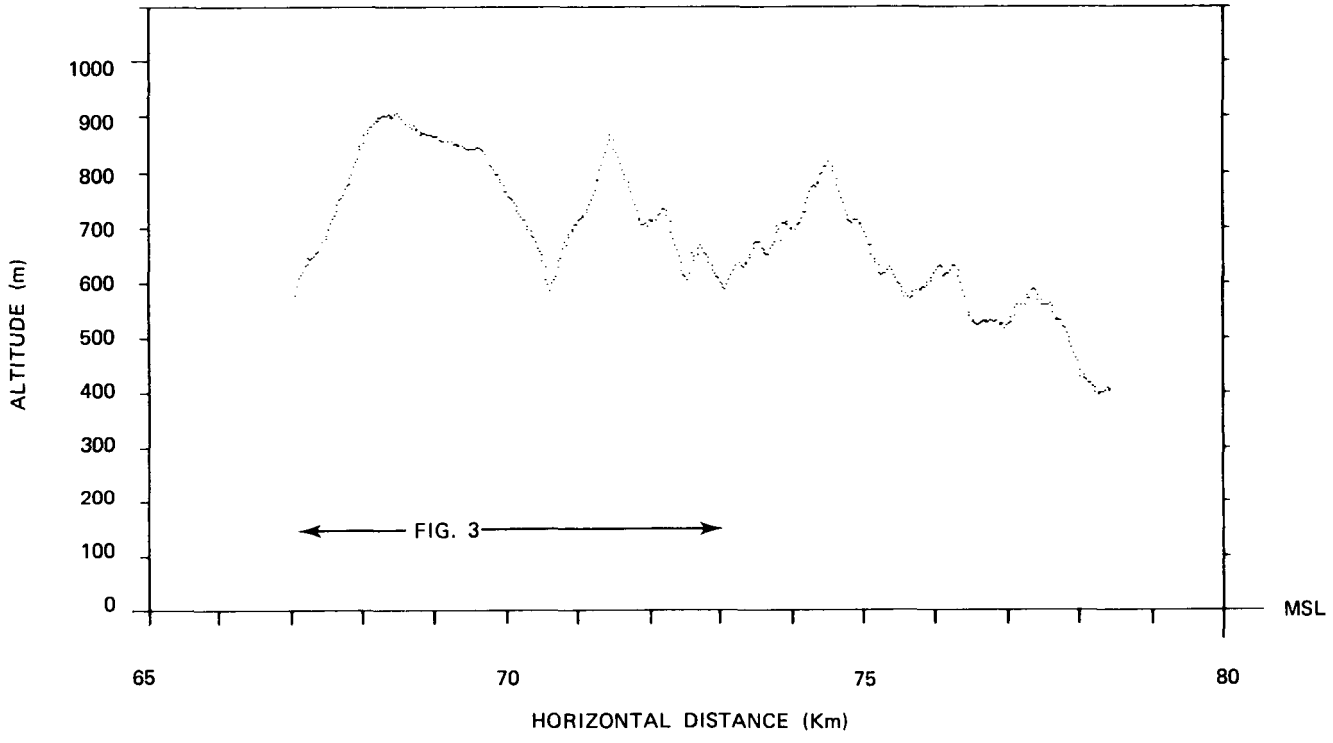

Figure 2. Subrange—Skyline Drive (Plot Shows 1 Data Record of 65 Seconds Duration, Consisting of 450 Laser Pulses)

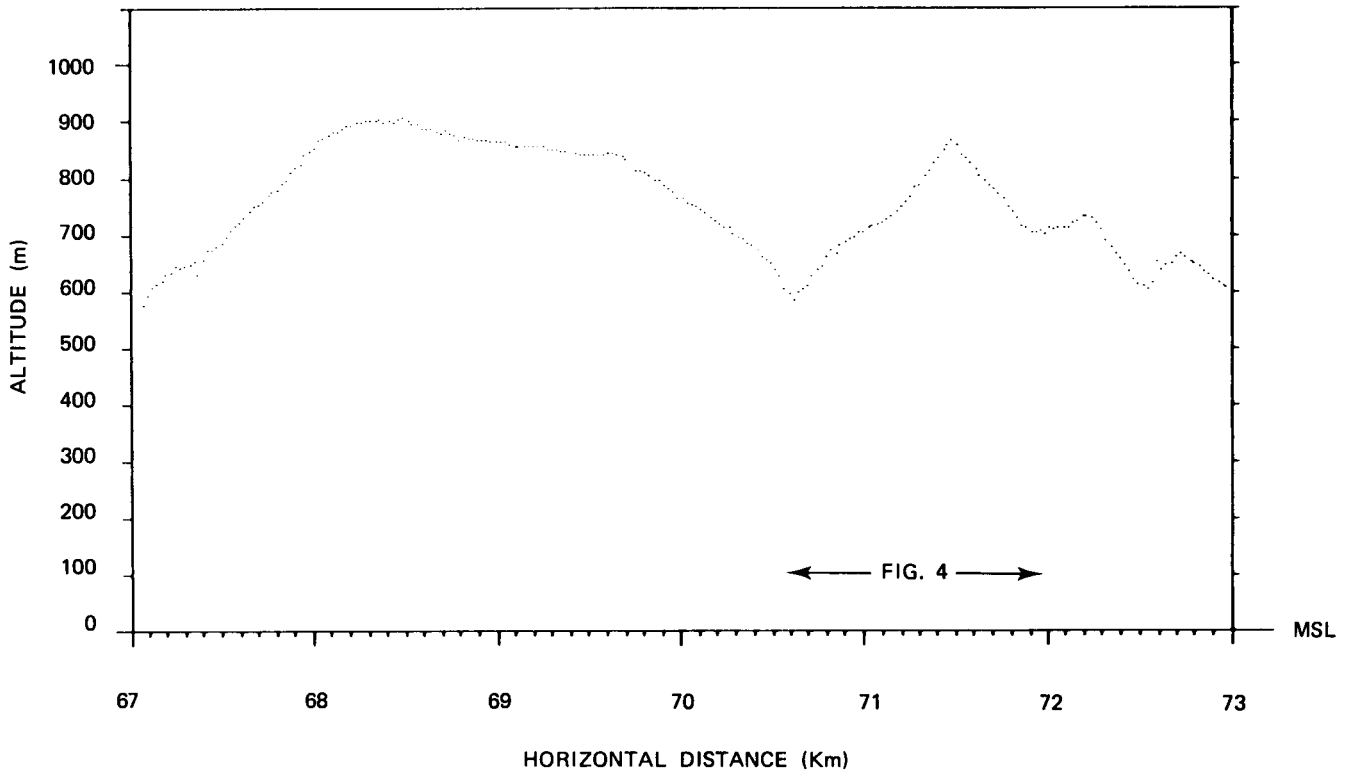

Figure 3. Subrange of Skyline Drive

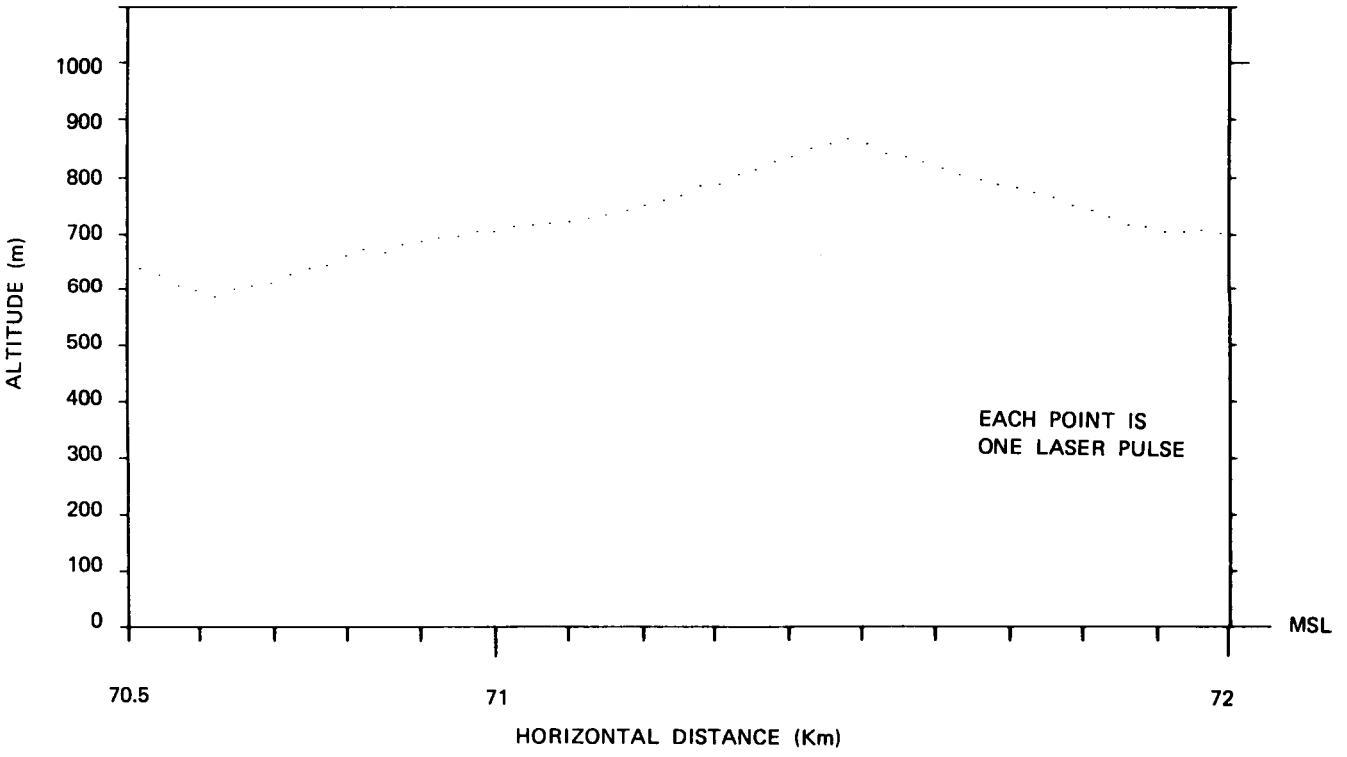

Figure 4. Subrange of Skyline Drive

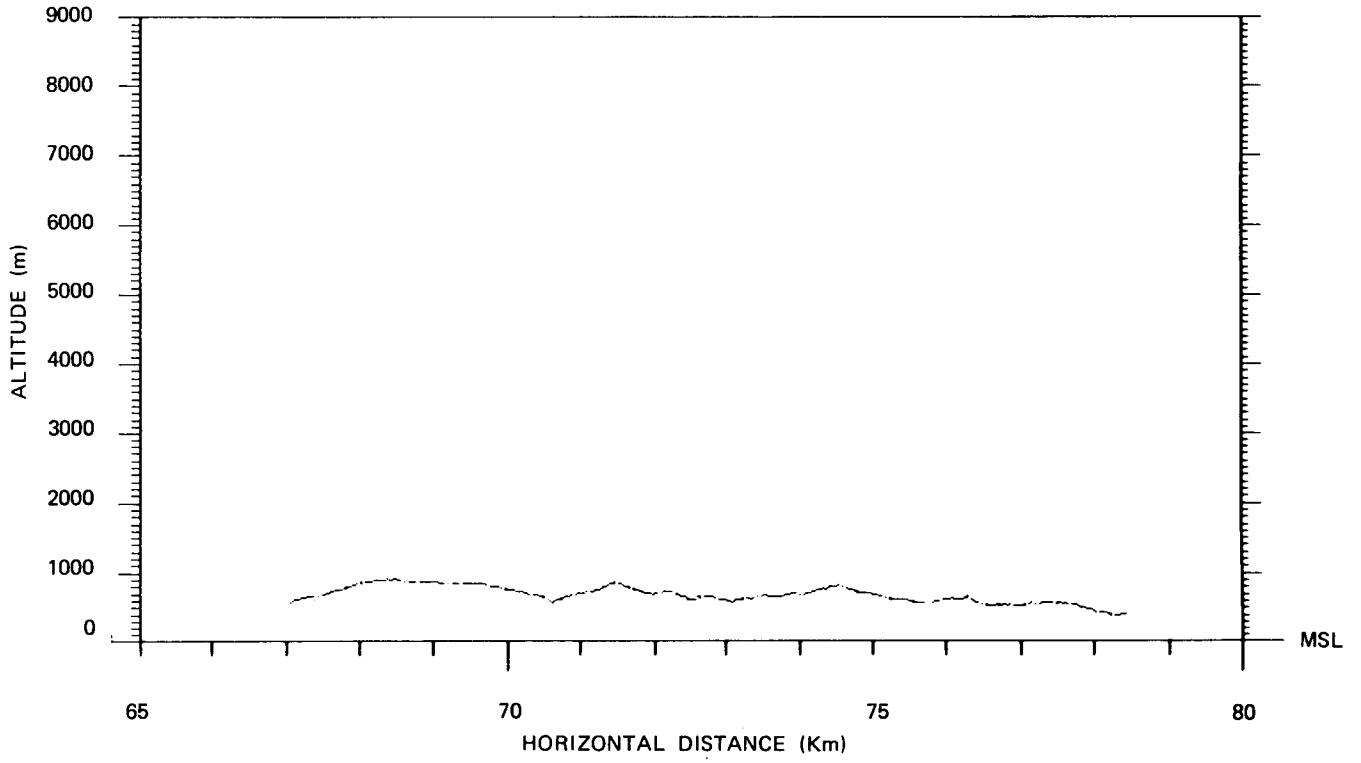

Figure 5. Same as Figure 2, but with the Two Axes on Nearly the Same Scale, to Give a Realistic View of the Terrain Profile.

PROGRAM LISTINGS

Preprocessing Routine

to Unpack Raw Data

```
      PROGRAM UNPACK
C     **************
C
C
C                  ALTIMETRY DATA TRANSFER PROGRAM
C                           PART II
C              UNPACK SLAP DATA DISK FILE INTO NIBBLES
C                        AND DO STATISTICS
C
C
C                           PGM:  ALTIM. FOR
C
C                           CREATED:  12/5/85
C
C                           EDITED:  12/27/85
C
C                           EDITED:  12/31/85  BY H. SAFREN
C
C
C       VARIABLES:  LDATA = LOGICAL*1 ARRAY OF RAW DATA PACKED
C                           IN NIBBLES FROM MAGNETIC TAPE
C                           CONTAINS 450 DATA BLOCKS OF 25 BYTES
C                   BDATA = LOGICAL*1 ARRAY OF ONE DATA BLOCK
C                           UNPACKED FROM NIBBLES INTO BYTES
C                      NB = NUMBER OF DATA BYTES
C                      NS = NUMBER OF DATA BLOCKS
C                      NN = NUMBER OF NIBBLES
C
C
      BYTE         LDATA(7200), BDATA(14400), INPUT(15), LRE(3), LTE(3),
     *             OUTPUT(14), QR2, LR(2), LS(2), AS(2), LTIME(10), GO
C
      INTEGER*2    TEM, REM, ATTEN, ITEST(25), SHOT, SMIN, SMAX, RANGE,
     *             RPLOT(45), IRE(3), ITE(3), IAS(2),
     *             MONTH, DAY, HOUR, MINUTE, SECOND, BEGFIL, ENDFIL,
     *             DISKNO, DISKNP
C
      EQUIVALENCE (RANGE,LR(1)), (SHOT,LS(1))
C
      DATA         INPUT/15*'000/, DISKNP/1/
C
C
C
C        OPEN FILE IN DLO TO HOLD UNPACKED DATA FILE, CREATED BY
C        UNPACKING AND CONCATENATING A SERIES OF RAW DATA FILES
C        ******************************************************
C
 1100    OPEN (UNIT=1, NAME='DLO:UFILE.DATA', TYPE='NEW', DISP='KEEP',
     *         FORM='FORMATTED', RECORDSIZE=24, INITIALSIZE=1000)
C
C
C        OPEN FILE WHICH LISTS THE SERIES OF RAW DATA FILES FOR THIS FLIGHT
C        ******************************************************************
C
         OPEN (UNIT=2, NAME='SCR:FILLST.RAW', TYPE='OLD', DISP='KEEP',
     *         FORM='FORMATTED', RECORDSIZE=16)
```

```
C         QUERY USER FOR FILES TO BE PROCESSED;
C         POSITION FILE LIST TO THE FIRST FILE SPECIFIED
C         ***********************************************
C
          TYPE 900
  900     FORMAT(//
     *     t5,'Type the sequence number of the first file to be processed: ',$)
          ACCEPT 901, BEGFIL
  901     FORMAT(I3)
C
          TYPE 910
  910     FORMAT(//
     *     t5,'Type the sequence number of the last file to be processed: ',$)
          ACCEPT 901, ENDFIL
C
          IF (BEGFIL .EQ. 1) GO TO 5
          DO 920 I=1,BEGFIL-1
          READ (2,921)  INPUT, DISKNO
          INPUT(15) = '000
  920     CONTINUE
  921     FORMAT(15A1,I1)
C
C         READ IN RAW (UNPACKED) DATA FILES FROM DISK(S)
C         **********************************************
C
     5    NB=7200
          NS=450
          NN=32
C
C
C         LOOP THROUGH THE SPECIFIED FILES
C         ********************************
C
          DO 1900  IFILE = 1, ENDFIL-BEGFIL+1
C         *******
C
          READ (2,1000)  INPUT, DISKNO
          INPUT(15) = '000
 1000     FORMAT(15A1, I1)
C
          IF (IFILE .EQ. 1)  GO TO 1005
          GO TO 1010
C
 1005     IF (DISKNO .EQ. 1)  GO TO 1020
          TYPE 1006, DISKNO
 1006     FORMAT(//
     *     T5,'Mount raw file disk number ',t33,I2,t36,';'/
     *     t5,'then type G, RETURN.'//)
          ACCEPT 1012, GO
          DISKNP = DISKNO
          GO TO 1020
C
 1010     IF (DISKNO .EQ. DISKNP)  GO TO 1020
```

```
      TYPE 1011
 1011 FORMAT(//t5,'Mount next raw data disk for this flight,'/
     *          t5,'then type G, RETURN:  ',$)
      ACCEPT 1012, GO
 1012 FORMAT(A1)
      DISKNP = DISKNO
C
 1020 OPEN (UNIT=8, NAME=INPUT, TYPE='OLD', DISP='KEEP',
     *       FORM='FORMATTED', RECORDSIZE=650
C
      DO 10 I=1,NS
      LMIN=1+(I-1)*16
      LMAX=LMIN+15
      READ(8,100) (LDATA(L),L=LMIN,LMAX)
   10 CONTINUE
  100 FORMAT(1X,16I4)
C
      CLOSE (UNIT=8, DISP='KEEP')
C
C
C     UNPACK NIBBLES INTO BYTES OF BDATA
C     **********************************
C
      RANGE=0                                  ! initialize variables
      ATTEN=0
      SHOT=0
      ZAVG=0.
      TAVG=0.
      RAVG=0.
      SZ=0.
      ST=0.
      SR=0.
      NZ=0
      NT=0
      NR=0
      K=1
      SMIN=1
      SMAX=450
C
C
      DO 30 I=1,NS
C     ************
      ITES=0
      IRES=0
C
      DO 20 J=1,16
      L=J+(I-1)*16
      IF(J.GE.11) GO TO 22
      ITEST(J)=LDATA(L)
      IF(ITEST(J).LT.0) LDATA(L)=LDATA(L)+128
   22 CONTINUE
      BDATA(K)=LDATA(L)-(LDATA(L)/16)*16
      K=K+1
      BDATA(K)=LDATA(L)/16
   20 K=K+1
```

```
      IF(ITEST(7).LT.0) ITES=80
      IF(ITEST(9).LT.0) IRES=80
      LR(2)=LDATA(1-4)
      LR(1)=LDATA(L-5)
      LS(1)=LDATA(L-1)
      LS(2)=LDATA(L)
      BDATA(K-6)=LDATA(L-2)-(LDATA(L-2)/16)*16
      BDATA(K-5)=LDATA(L-2)/16
      AS(1)=BDATA(K-6)
      AS(2)=BDATA(K-5)
      IAS(1)=AS(1)
      IAS(2)=AS(2)
      LRE(1)=BDATA(K-14)
      LRE(2)=BDATA(K-15)
      LRE(3)=BDATA(K-16)
      LTE(1)=BDATA(K-18)
      LTE(2)=BDATA(K-19)
      LTE(3)=BDATA(K-20)
C
        DO 25 J=1,3
        IRE(J)=LRE(J)
        ITE(J)=LTE(J)
25      CONTINUE
C
      REM=IRE(1)*100+IRE(2)*10+IRE(3)+IRES
      TEM=ITE(1)*100+ITE(2)*10+ITE(3)+ITES
      ATTEN=IAS(1)*10+IAS(2)
      LTIME(1)=BDATA(K-21)
      LTIME(2)=BDATA(K-22)
      LTIME(3)=BDATA(K-23)
      LTIME(4)=BDATA(K-24)
      LTIME(5)=BDATA(K-27)
      LTIME(6)=BDATA(K-28)
      LTIME(7)=BDATA(K-29)
      LTIME(8)=BDATA(K-30)
      LTIME(9)=BDATA(K-31)
      LTIME(10)=BDATA(K-32)
      IF(TEM.NE.0) NT=NT+1
      IF(REM.NE.0) NR=NR+1
      IF(RANGE.NE.0) NZ=NZ+1
      Z=FLOAT(RANGE)
      R=FLOAT(REM)
      T=FLOAT(TEM)
      ZAVG=ZAVG+Z
      TAVG=TAVG+T
      RAVG=RAVG+R
      SZ=SZ+Z**2
      ST=ST+T**2
      SR=SR+R**2
```

```
          LMIN = 1 + (I-1)*32
          LMAX = LMIN + 31
C                                                          ! type first ten records
C         IF (I .LE. 10) TYPE 400, (BDATA(L),L=LMIN,LMAX)
  400     FORMAT(' '/2X,16(1X,I3)/2X,16(1X,I3))
          IF (I .LE. 10) TYPE 520, (LTIME(J),J=1,10),RANGE,REM,TEM,ATTEN,SHOT
  520     FORMAT(2X,5(2I1,1X),2X,I9,2X,I4,2X,I4,2X,I3,2X,I6)
C
C                                                          ! write current record to
C                                                          ! unpacked file in DL0
          MONTH = LTIME(1)*10 + LTIME(2)
          DAY   = LTIME(3)*10 + LTIME(4)
          HOUR  = LTIME(5)*10 + LTIME(6)
          MINUTE= LTIME(7)*10 + LTIME(8)
          SECOND= LTIME(9)*10 + LTIME(10)
          WRITE (1,521)   MONTH, DAY, HOUR, MINUTE, SECOND, RANGE
  521     FORMAT(5I3,I9)
C
          IF ((SHOT .LT. SMIN)  .AND.  (SHOT .GT. SMAX))  GO TO 30
C
    30    CONTINUE
C         ********
C
 1900     CONTINUE                                         ! end of loop through files
C         ********
C
C
C         CLOSE RUN
C         *********
C
          CLOSE (UNIT=2, DISP='KEEP')
          CLOSE (UNIT=1, DISP='KEEP')
C
          STOP
          END
```

Processing Routine

```
      PROGRAM PROCESS
C
C  This routine processes the unpacked altimeter data.
C*************************************************
C
      BYTE          REPLY
C
      INTEGER*2     MONTH(450), DAY(450),
     *              HOUR(450), MINUTE(450), SECOND(450),
     *              RETTIM(450), GROUPN, HR1, GOODCT
C
      REAL*4        MSLHT(450), RTSEC(450), HRANGE(450), TIMSEC(450)
C
      DATA          REFALT/0./, SPEED/175.0/, C/2.997925E8/, RINDEX/1.0/,
     *              GROUPN/0/,  GOODCT/0/
C
C*************************************************************************
C
C     PROCESS UNPACKED DATA FILE; CREATE PROCESSED FILE TO BE PLOTTED
C     ***************************************************************
C
C  This section of code processes the unpacked data file and stores the
C  resulting file, which is ready for plotting, in DL0.
C
C
C       Open the unpacked data file in DL0
C       **********************************
C
    1 OPEN (UNIT=1, NAME='DL0:UFILE.DAT', TYPE='OLD', DISP='DELETE',
     *       FORM='FORMATTED', RECORDSIZE=24)
C
C
C       Open a file in DL0 to hold the processed file
C       *********************************************
C
      OPEN (UNIT=2, NAME='DL0:PROFIL.DAT', TYPE='NEW', DISP='KEEP',
     *       FORM='FORMATTED', RECORDSIZE=26, INITIALSIZE=1000)
C
C       Query user for values of parameters
C       ***********************************
C
   30 TYPE 31, REFALT
   31 FORMAT(//
     * t2,'The current value of the altitude (above mean sea level)'/
     * t2,'of the starting point is:',t28,f9.2,t39,'meters;'/
     * t2,'do you want to change this? (Y or N):  ',$)
      ACCEPT 11, REPLY
   11 FORMAT(A1)
      IF (REPLY .EQ. 'N') GO TO 40
      TYPE 32
   32 FORMAT(/T2,'Type the altitude (above mean sea level)'/
     *        t2,'of the starting point, in meters:  ',$)
      ACCEPT 33, REFALT
   33 FORMAT(f12.4)
```

```
   40     TYPE 41,  SPEED
   41     FORMAT(//T2,'The current value of the airplane speed is:  ',
     *             t46,f7.2,t54,'meters per second;'/
     *             t2,'do you wnat to change this? (Y or N):  ',$)
          ACCEPT 11, REPLY
          IF (REPLY .EQ. 'N')  GO TO 50
          TYPE 42
   42     FORMAT(/T2,'Type the new speed (in meters per second):  ',$)
          ACCEPT 43, SPEED
   43     FORMAT(f8.2)
C
   50     TYPE 51, RINDEX
   51     FORMAT(//
     *    t2,'The value of the average index of refraction of the air'/
     *    t2,'along the vertical path is taken to be:',t42,f10.6,';'/
     *  t2,'do you want to replace this by a more precise value? (Y or N): ',$)
          ACCEPT 11, REPLY
          VLIGHT = C/RINDEX
          IF (REPLY .EQ. 'N')  GO TO 1319
          TYPE 52
   52     FORMAT(/T2,'Type the new value:  ',$)
          ACCEPT 53, RINDEX
   53     FORMAT(G13.6)
          VLIGHT = C/RINDEX              ! effective velocity of light in air
C                                          (averaged over vertical path)
C
C
C         Write these parameter values to the processed file
C         ***************************************************
C
 1319     WRITE (2, 1320)  REFALT
 1320     FORMAT(F12.4)
          WRITE (2, 1321)  SPEED
 1321     FORMAT(F8.2)
          WRITE(2, 1322)  RINDEX
 1322     FORMAT(G13.6)
C
C
C         Read a group of 450 data points from the unpacked file in DL0;
C         store them in arrays.
C         ************************************************************
C
  100     DO 110  I=1,450
          READ (1,111, END=499)  MONTH(I), DAY(I),
     *                           HOUR(I), MINUTE(I), SECOND(I),
     *                           RETTIM(I)
  110     CONTINUE
  111     FORMAT(5I3,I9)
          GROUPN = GROUPN + 1
```

```
C        Adjust the times for the 450 data points
C        (the times are given only to the preceding second)
C        ************************************************
C
C        Adjust hour(1) to 0 and take succeeding hours relative to this
C        --------------------------------------------------------------
         HR1 = HOUR(1)
         DO 120  I=1,450
         HOUR(I) = HOUR(I) - HOUR(1)
  120    CONTINUE
C
C
C        Convert (integer) times in hours, minutes and seconds
C        to real times in seconds
C        -------------------------------------------------
         DO 130  I=1,450
         TIMSEC (I) =    FLOAT(HOUR(I))*3600.
     *                 + FLOAT(MINUTE(I))*GO.
     *                 + FLOAT(SECOND(I))
  130    CONTINUE
C
C
C        Find the first changed time; assume this is exact
C        and use it as the reference time for this group of 450 points
C        ------------------------------------------------------------
         INDEX = 2
  160    IF (SECOND(INDEX) .NE. SECOND(1))  GO TO 161
         INDEX = INDEX + 1
         GO TO 160
  161    INDREF = INDEX
         REFTIM = TIMSEC(INDREF)
C
C
C        Find the last changed time; assume temporarily that it is exact
C        ---------------------------------------------------------------
         INDEX = 449
  163    IF (SECOND(INDEX) .NE. SECOND(450))  GO TO 164
         INDEX = INDEX - 1
         GO TO 163
  164    INDLST = INDEX + 1
         REFTM2 = TIMSEC(INDLST)
C
C
C        Get more accurate estimates for the first and last times,
C        using the two reference times and assuming 7 data points per second;
C        compute the time between data points.
C        ------------------------------------------------------------------
         RTSEC(1)   = REFTIM - FLOAT(INDREF-1)  *(1./7.)
         RTSEC(450) = REFTM2 + FLOAT(450-INDLST)*(1./7.)
         DELTIM     = (RTSEC(450) - RTSEC(1)) / 449.
```

```
C        Use this time interval to compute (real) times for the data points
C        that are accurate to better than the preceding second
C        ------------------------------------------------------------------
         DO 170  I=1,450
         RTSEC(I) = REFTIM + FLOAT(I-INDREF)*DELTIM
  170    CONTINUE
C
C
C        Add back in hour(1), to get the actual (real) time in seconds;
C        then subtract hour(1) for the first group of 450 points to get
C        time in seconds relative to the first data point in the unpacked file
C        ------------------------------------------------------------------
         IF (GROUPN .EQ. 1)  TINIT = RTSEC(1) + FLOAT(HR1)*3600.
C
         DO 180  I=1,450
         RTSEC(I) = RTSEC(I) + FLOAT(HR1)*3600. - TINIT
  180    CONTINUE
C
C
C        Convert the data point times into distances in kilometers
C        from the starting point
C        ********************************************************
C
         DO 190  I=1,450
         HRANGE(I) = RTSEC(I) * SPEED/1000.          ! distance in km
  190    CONTINUE
C
C
C        Convert the return times (in tens of nanoseconds)
C        to heights of the ground above mean sea level (in meters)
C        ********************************************************
C
         DO 200  I=1,450
         ALT = 5.E-9 * VLIGHT * FLOAT(RETTIM(I)) ! altitude of airplane above
C                                                  local ground (in meters)
         IF (I .EQ. 1  .AND.                     ! save altitude at starting
     *       GROUPN .EQ. 1)  ALTO = ALT          ! point, for reference
         RELHT = ALTO - ALT                      ! ground height above
C                                                  starting point (meters)
         MSLHT(I) = RELHT + REFALT               ! ground height above mean
C                                                  sea level (meters)
  200    CONTINUE
C
C
C        Write this group of processed data points to the processed file;
C        do not write "bad" points (return time is zero).
C        ********************************************************
C
         DO 250  I=1,450
         II = I
         IF (RETTIM(II) .EQ. 0)  GO TO 250
         WRITE (2,1351) HRANGE(II), MSLHT(II)
         GOODCT = GOODCT + 1
  250    CONTINUE
 1351    FORMAT(F14.5, F12.2)
```

```
C         Display the time between data points for this group,
C         the inter-group time gap from the last group and
C         the current total number of good points (i.e., the
C         current number of points in the processed file)
C         ************************************************
C
          IF (GROUPN .EQ. 1)  ENDTPR = 0.
          GAP = RTSEC(1) - ENDTPR
          ENDTPR = RTSEC(450)
C
          TYPE 260,  DELTIM, GAP, GOODCT
  260     FORMAT(////
     *     t5,'Time between data points for this group = ',t48,f7.4,
     *                                                  t56,'seconds;'/
     *     t5,'Gap from last group = ',t28,f8.4,t37,'seconds;'//
     *     t5,'Current number of points in processed file = ',t47,I6)
C
C
C         Read next group of 450 points from the unpacked data file
C         ************************************************************
          GO TO 100
C
C*************************************************************************
C
C                         CLOSE RUN
C                         *********
C
  499     CLOSE (UNIT=1, DISP='DELETE')
          CLOSE (UNIT=2, DISP='KEEP')
          STOP
          END
```

Plot Routine

```
      PROGRAM PLOT
C
C   This routine plots the processed altimeter data.
C*************************************************************
C
      BYTE          REPLY, LINEAR(10), STDSCR(10),
     *              LINE, BRIGHT, DOTTED(2), GO, USER, DARK,
     *              YES, NO, STRING(14), LFTORD(2), LONG, MEDIUM,
     *              BOTTOM(2), NORLIN(2), BEGNUM(10), ENDNUM(10),
     *              HSCALE, STDRD, RGTORD(2), PROFIL(15), SHORT,
     *              LFTNUM(5), COMAND(14)
C
      INTEGER*2     IPARAM(10), GOODCT, REPLOT(3)
C
      REAL*4        RPARAM(10),  USRWIN(4), SCRWIN(4)
C
      COMMON/COORDS/  UX,UY, TX,TY, SX,SY
C
      DATA          LINEAR/'L','I','N',7*' '/,
     *              STDSCR/'S',9*' '/, LINE/'L'/, BRIGHT/'B'/,
     *              DOTTED/'D','T'/, USER/'U'/, DARK/'D'/,
     *              YES/'Y'/, NO/'N'/,
     *              STRING/'s','e','a','r','c','h','i','n','g',' ',
     *                     'f','i','l','e'/,
     *              LFTORD/'O','L'/, LONG/'L'/, MEDIUM/'M'/,
     *              BOTTOM/'A','B'/, NORLIN/'N','R'/,
     *              BEGNUM/12*' '/, ENDNUM/12*' '/, STDRD/'S'/,
     *              RGTORD/'O','R'/, PROFIL/15*000/, SHORT/'S'/,
     *              LFTNUM/5*' '/,
     *              COMAND/'@','C','O','M',':','R','P','L','O','T','`',
     *                     'C','O','M'/
C*************************************************************************
C
C                COPY THE PROCESSED FILE INTO VM
C                (in unformatted form)
C                ******************************
C
C
C       Open the file to be plotted
C       ---------------------------
      OPEN (UNIT=1, NAME='DL0:PROFIL.DAT', TYPE='OLD', DISP='DELETE',
     *       FORM='FORMATTED', RECORDSIZE=26)
C
C
C       Read the first three records; these contain the parameter values
C       used when the file was created
C       ------------------------------------------------------------
      READ (1, 2120)  REFALT
 2120 FORMAT(F12.4)
      READ (1, 2121)  SPEED
 2121 FORMAT(F8.2)
      READ (1, 2122)  RINDEX
 2122 FORMAT(G13.6)
```

```
C         Display these parameter values
C         ------------------------------
C
          TYPE 2125,  REFALT, SPEED, RINDEX
 2125     FORMAT(///
     *     t5,'Values of parameters used when this file was created:'//
     *      t10,'Height of starting point above mean sea level (m) = ',
     *                                                        t63,f12.4//
     *      t10,'Speed of airplane (m/sec) = ',t39,f8.2//
     *      t10,'Effective index of refraction = ',t43,g13.6///)
C
C
C         Open file in VM (to hold unformatted version of the file)
C         ---------------------------------------------------------
C
          OPEN (UNIT=2, NAME='VM:PLTFIL.DAT', TYPE='NEW', DISP='DELETE',
     *          FORM='UNFORMATTED', ACCESS='DIRECT', MAXREC=20480,
     *          RECORDSIZE=2, INITIALSIZE=320)
C
C
C         Copy file to VM
C         ---------------
C
          TYPE 2130
 2130     FORMAT(////T20,'Copying processed file in DL0 to VM ...'/
     *               t20,'(direct access, unformatted)'//)
C
          IVM = 1
C
 2140     READ (1, 2150, END=2200)  RANGE,SLHT
 2150     FORMAT(F14.5, F12.2)
C
          WRITE (2'IVM)  RANGE, SLHT
C
          IVM = IVM + 1
          GO TO 2140
C
 2200     GOODCT = IVM - 1
          CLOSE (UNIT=1, DISP='DELETE')
C
C
C                          PLOT PROCESSED DATA
C                          *******************
C
C         Set up plotting system
C         **********************
          BEGRNG = 0.
          ENDRNG = 10000.
          HSCALE = STDRD
C
  500     CALL BEGPLT
```

```
C       Search processed data file to find maximum height and range
C       ***********************************************************
        HTMAX = -1000.
        HTMIN =  100000.
C
        DO 550  I=1,GOODCT
        II = I
        READ (2'II)  RANGE, HEIGHT
        RNGMAX = RANGE
        IF (HEIGHT .EQ. -1.E10)  GO TO 550
        IF (RANGE .LT. BEGRNG)   GO TO 550
        IF (RANGE .GT. ENDRNG)   GO TO 550
        IF (HEIGHT .LT. HTMIN)  HTMIN = HEIGHT
        IF (HEIGHT .GT. HTMAX)  HTMAX = HEIGHT
  550   CONTINUE
C
  560   IF (ENDRNG .EQ. 10000.)  ENDRNG = RNGMAX
C
C
C       Set up the mapping from user space to the screen
C       ************************************************
C
        USRWIN(1) = BEGRNG                          ! user window
        USRWIN(2) = ENDRNG
C
        SBOTTM    = AMIN1 (HTMIN-100., 0.)
        STOP      = AMAX1(HTMAX+1000., 0.)
C
        IF (HSCALE .EQ. 'S') GO TO 561
        IF (HSCALE .EQ. 'D') GO TO 567
        IF (HSCALE .EQ. 'K') GO TO 567
C
  561   USRWIN(3) = SBOTTM
        USRWIN(4) = STOP
        CBOTTM    = SBOTTM
        CTOP      = STOP
        GO TO 570
C
  567   USRWIN(3) = CBOTTM
        USRWIN(4) = CTOP
C
  570   CALL SETMAP (LINEAR, STDSCR, IPARAM, RPARAM, USRWIN, SCRWIN)
C
C
C       Draw a box around the screen window
C       ***********************************
        CALL BOXWIN
C
C
C       Draw a horizontal dotted line to show mean sea level
C       ****************************************************
        CALL DRWLIN (0.,0., ENDRNG,0., LINE, BRIGHT, DOTTED, IPARAM)
```

```
C       Plot the entire set of processed points
C       ***************************************
C
        DO 600  I=1,GOODCT
        II = I
        READ (2'II)  RANGE, HEIGHT
        IF (HEIGHT .EQ. -1.E10)  GO TO 600
        IF (RANGE .LT. BEGRNG)   GO TO 600
        IF (RANGE .GT. ENDRNG)   GO TO 600
        CALL PLOTPT (RANGE, HEIGHT, BRIGHT)
  600   CONTINUE
C
C
C       Draw tick marks every 100 meters on the vertical axis;
C       display height every 100 meters, if top < 1,000 meters
C       ******************************************************
C
        CALL TICK (LFTORD, LONG, 0., BRIGHT)     ! long tick at sea level
        CALL TICK (RGTORD, LONG, 0., BRIGHT)
C
        TCKHT = 0.                               ! short ticks above sea
  610   TCKHT = TCKHT + 100.                     ! level, every 100 meters
        IF (TCKHT .GT. USRWIN(4))  GO TO 620
        CALL TICK (LFTORD, SHORT, TCKHT, BRIGHT)
        CALL TICK (RGTORD, SHORT, TCKHT, BRIGHT)
C
        IF (USRWIN(4) .GT. 1000.)  GO TO 610     ! display height every 100
C        ENCODE (8, 611, LFTNUM)  TCKHT          ! meters
C 611    FORMAT(F8.1)
        UX = 0.
        UY = TCKHT
        CALL MAP (USER)
        ITCKHT = SY
        ENCODE (5, 611, LFTNUM)  IFIX(TCKHT)
  611   FORMAT(I5)
        CALL DISPSQ (LFTNUM, 5, BRIGHT, 1, 0, ITCKHT, 0., 15)
        GO TO 610
C
  620   TCKHT = 0.                               ! medium ticks every 1000
  621   TCKHT = TCKHT + 1000.                    ! meters; display heights
        IF (TCKHT .GT. USRWIN(4))  GO TO 630
        CALL TICK (LFTORD, MEDIUM, TCKHT, BRIGHT)
        CALL TICK (RGTORD, MEDIUM, TCKHT, BRIGHT)
C        ENCODE (8, 611, LFTNUM)  TCKHT
        UX = 0.
        UY = TCKHT
        CALL MAP (USER)
        ITCKHT = SY
        ENCODE (5, 611, LFTNUM)  IFIX(TCKHT)
        CALL DISPSQ (LFTNUM, 5, BRIGHT, 1, 0, ITCKHT, 0., 15)
        GO TO 621
C
  630   TCKHT = 0.                               ! short ticks below sea
  631   TCKHT = TCKHT - 100.                     ! level, every 100 meters
        IF (TCKHT .LT. USRWIN(3))  GO TO 635
        CALL TICK (LFTORD, SHORT, TCKHT, BRIGHT)
        CALL TICK (RGTORD, SHORT, TCKHT, BRIGHT)
        GO TO 631
```

```
  635     TCKHT = 0.                                    ! medium ticks below sea
  636     TCKHT = TCKHT - 1000.                         ! level, every 1,000 meters
          IF (TCKHT .LT. USRWIN(3)) GO TO 640
          CALL TICK (LFTORD, MEDIUM, TCKHT, BRIGHT)
          CALL TICK (RGTORD, MEDIUM, TCKHT, BRIGHT)
          GO TO 636
C
C
C         Find appropriate range unit
C         (at least two units in range span)
C         **********************************
C
  640     RSPAN = ENDRNG - BEGRNG
          TESTU = 100000.
C
  641     TESTU = TESTU/10.
          IF (TESTU .GT. 0.5*RSPAN)  GO TO 641
          RUNIT = TESTU
C
C
C         Draw range ticks every unit
C         ***************************
C
          QUOT   = BEGRNG/RUNIT
          TRQUOT = FLOAT (IFIX(QUOT))
          REM    = QUOT - TRQUOT
C
          BEGTCK = TRQUOT*RUNIT + RUNIT
          IF (REM .EQ. 0.)  BEGTCK = TRQUOT*RUNIT
C
          TCKP = BEGTCK - RUNIT
  651     TCKP = TCKP + RUNIT
          IF (TCKP .GT. USRWIN(2))  GO TO 652
          CALL TICK (BOTTOM, MEDIUM, TCKP, BRIGHT)
          QUOT   = TCKP/(10.*RUNIT)
          TRQUOT = FLOAT (IFIX(QUOT))
          DIFF   = QUOT - TRQUOT
          IF (DIFF .LT. 0.01  .OR.  DIFF .GT. 0.99)
     *        CALL TICK (BOTTOM, LONG, TCKP, BRIGHT)
          GO TO 651
  652     ENDTCK = TCKP - RUNIT
C
C
C         Draw ticks every 1/10th range unit, if there are
C         no more than ten units
C         ***********************************************
C
  660     IF (IFIX (RSPAN/RUNIT) .GT. 10)  GO TO 670
C
          TCKP = BEGTKP - RUNIT
          TENTH = RUNIT/10.
C
  661     TCKP = TCKP + TENTH
          IF (TCKP .LT. USRWIN(1))  GO TO 661
          IF (TCKP .GT. USRWIN(2))  GO TO 670
          CALL TICK (BOTTOM, SHORT, TCKP, BRIGHT)
          GO TO 661
```

```
C         Display range values at beginning and end (long) ticks
C         ******************************************************
C
  670     ENCODE (10, 671, BEGNUM)  BEGTCK
          ENCODE (10, 671, ENDNUM)  ENDTCK
  671     FORMAT(F10.3)
C
          UX = BEGTCK
          UY = 0.
          CALL MAP (USER)
          IBEGTK  = SX
C
          UX = ENDTCK
          UY = 0.
          CALL MAP (USER)
          IENDTK = SX
C
          CALL DISPSQ (BEGNUM,10,BRIGHT,1,IBEGTK-75,0,0.,15)
          CALL DISPSQ (ENDNUM,10,BRIGHT,1,IENDTK-75,0,0.,15)
C
          ACCEPT 680, GO                          ! Hold plot on screen
  680     FORMAT(A1)                              ! until user's cue
C
C
C                   REMOVE SPURIOUS POINTS FROM PROCESSED
C                   DATA FILE, INTERACTIVELY, BY INSPECTION
C                   ***************************************
C
C         Query user about removing spurious points
C         *****************************************
C
          CALL ECHO (YES)
          CALL TYP100
          TYPE 700
  700     FORMAT(////T10,'Do you want to interactively remove any'/
     *             t10,'spurious points from this plot (and from'/
     *             t10,'the processed data file)? (Y or N):  ',$)
          ACCEPT 701, REPLY
  701     FORMAT(A1)
          CALL ECHO (NO)
          CALL CLR100
          IF (REPLY .EQ. 'Y')  GO TO 705
          IF (REPLY .EQ. 'N')  GO TO 706
  705     REPLOT(1) = 1
          GO TO 800
  706     REPLOT(1) = 0
          GO TO 900
```

```
C       Remove spurious points by using the crosshair
C       ********************************************
C
  800   CALL ECHO (YES)
        CALL TYP100
        TYPE 801
  801   FORMAT(///t5,'Use the crosshair to remove unwanted points:'//
     *          t10,'o  Move the crosshair around the screen by using the'/
     *          t10,'   four arrow keys;'//
     *          t10,'o  To remove a point, position the crosshair NEAR the'/
     *          t10,'   point (not exactly on it) and type period, RETURN;'//
     *          t10,'o  To recall the crosshair to remove another point,'/
     *          t10,'   type N (no RETURN);'//
     *          t10,'o  After removing the last undesired point,'/
     *          t10,'   type G (no RETURN).'///
     *          t5,'    To remove this message and call up the crosshair,'/
     *          t10,'   type G, RETURN.')
        ACCEPT 802, GO
  802   FORMAT(A1)
        CALL ECHO (NO)
        CALL CLR100
C
        IXINIT = 500
        IYINIT = 650
  810   CALL MOVXHR (IXINIT,IYINIT, IXPOS,IYPOS)            ! crosshair mode
C
        CALL DISPST (STRING, 14, BRIGHT, 1, 10,750)
C
        MINDST = 10000
        DO 830  I=1,GOODCT
C       ******************
        II = I
        READ (2'II)  RANGE, SLHT
        IF (SLHT .EQ. -1.E10)  GO TO 830
        IF (RANGE .LT. BEGRNG) GO TO 830
        IF (RANGE .GT. ENDRNG) GO TO 830
C
        UX = RANGE
        UY = SLHT
        CALL MAP (USER)
        IHRNGE = SX
        IMSLHT = SY
C
        IDIST = IABS(IHRNGE-IXPOS) + IABS(IMSLHT-IYPOS)
        IF (IDIST .GE. MINDST)  GO TO 830
        MINDST = IDIST
        IMIN   = II
        HRNGMN = RANGE
        SLHMN  = SLHT
  830   CONTINUE
C       ********
C
        WRITE (2'IMIN)  HRNGMN, -1.E10     ! set height of bad point
C                                          ! to large negative value
C
        CALL PLOTPT (HRNGMN,SLHMN, DARK)   ! erase bad point from screen
        CALL DISPST (STRING, 14, DARK, 1, 10,750)
```

```
  840     ICHAR = ITTINR()
          IF (ICHAR .EQ. 78)  GO TO 850       ! get next bad point
          GO TO 860
  850     IXINIT = IXPOS
          IYINIT = IYPOS
          GO TO 810
  860     IF (ICHAR .EQ. 71)  GO TO 900       ! go to next section of code
          GO TO 840
C
C
C                        EXPAND HORIZONTAL SCALE - CHOOSE A
C                        RANGE INTERVAL AND PLOT IT
C                        **********************************
C
C         Query user about plotting a subrange
C         ------------------------------------
C
  900     CALL ECHO (YES)
          CALL TYP100
          TYPE 901
  901     FORMAT(///
     *     t10,'Do you want to plot a subrange of the data? (Y or N):  ',$)
          ACCEPT 902, REPLY
  902     FORMAT(A1)
          IF (REPLY .EQ. 'N')  GO TO 910
          IF (REPLY .EQ. 'Y')  GO TO 911
  910     REPLOT(2) = 0
          GO TO 903
  911     REPLOT(2) = 1
          GO TO 905
  903     CALL ECHO (NO)
          CALL CLR100
          GO TO 1000
C
  905     TYPE 906
  906     FORMAT(//
     *     t10,'Do you want to specify the subrange by Typing the endpoints'/
     *     t10,'or by using the Crosshair? (type T or C):  ',$)
  907     ACCEPT 908, REPLY
  908     FORMAT(A1)
          CALL ECHO (NO)
          CALL CLR100
          IF (REPLY .EQ. 'T')  GO TO 920
          IF (REPLY .EQ. 'C')  GO TO 930
          GO TO 907
```

```
C         Specify subrange by typing endpoints
C         ------------------------------------
C
  920     CALL ECHO (YES)
          CALL TYP100
          TYPE 921
  921     FORMAT(///
     *     t10,'Type left endpoint of subrange (real, in km):  ',$)
          ACCEPT 922, BEGRNG
  922     FORMAT(G16.4)
          TYPE 923
  923     FORMAT(///
     *     t10,'Type right endpoint of subrange (real, in km):  ',$)
          ACCEPT 922, ENDRNG
          CALL ECHO (NO)
          CALL CLR100
          GO TO 1000
C
C
C         Specify subrange by using crosshair
C         -----------------------------------
C
  930     CALL ECHO (YES)
          CALL TYP100
          TYPE 931
  931     FORMAT(///
     *     t5, 'Use the crosshair to point to the endpoints of the subrange:'//
     *     t10,'o   Move the crosshair around the screen by using the'/
     *     t10,'    four arrow keys;'//
     *     t10,'o   Point to the left endpoint first,'/
     *     t10,'    then to the right endpoint;'//
     *     t10,'o   After positioning the crosshair at each endpoint,'/
     *     t10,'    type a period followed by RETURN.'//
     *     t5, 'To remove this message and call up the crosshair,'/
     *     t5, 'type G, RETURN.')
          ACCEPT 932, GO
  932     FORMAT(A1)
          CALL ECHO (NO)
          CALL CLR100
C
          CALL MOVXHR (500,0, IXPOS,IYPOS)          ! crosshair mode
          BEGRNG = BEGRNG + ((ENDRNG-BEGRNG)*(FLOAT(IXPOS)-100.))/823.
          CALL VECTOR (FLOAT(IXPOS),36., FLOAT(IXPOS),76.,
     *                 BRIGHT, NORLIN, IPARAM)
          CALL VECTOR (FLOAT(IXPOS-1),36., FLOAT(IXPOS-1),76.,
     *                 BRIGHT, NORLIN, IPARAM)
C
          CALL MOVXHR (IXPOS,IYPOS, IXPOS,IYPOS)
          ENDRNG = BEGRNG + ((ENDRNG-BEGRNG)*(FLOAT(IXPOS)-100.))/823.
          CALL VECTOR (FLOAT(IXPOS),36., FLOAT(IXPOS),76.,
     *                 BRIGHT, NORLIN, IPARAM)
          CALL VECTOR (FLOAT(IXPOS+1),36., FLOAT(IXPOS+1),76.,
     *                 BRIGHT, NORLIN, IPARAM)
          ACCEPT 940, GO
  940     FORMAT(A1)
```

```
C                       CHOOSE VERTICAL SCALE
C                       *********************
C
 1000     CALL ECHO (YES)
          CALL TYP100
          TYPE 1001
 1001     FORMAT(///
     *     t5,'Do you wnat to re-define the vertical scale? (Y or N):  ',$)
          ACCEPT 1002, REPLY
 1002     FORMAT(A1)
          IF (REPLY .EQ. 'Y')  GO TO 1010
          IF (REPLY .EQ. 'N')  GO TO 1005
C
 1005     REPLOT(3) = 0
          CALL ECHO (NO)
          CALL CLR100
          GO TO 1100
C
 1010     REPLOT(3) = 1
          TYPE 1011, CBOTTM, CTOP
 1011     FORMAT(///
     *     t5,'Before replotting the data, you may choose the vertical scale:'//
     *     t10,'S.   The standard vertical scale extends from 100 meters below'/
     *     t10,'     the lowest data point or zero (sea level), whichever is'/
     *     t10,'     less, to 1,000 meters above the highest data point;'/
     *     t10,'     this option MUST be chosen if you want to display all'/
     *     t10,'     the outlying points in order to remove them;'//
     *     t10,'D.   You may define a new vertical scale;'//
     *     t10,'K.   You may keep the current vertical scale, which is:'/
     *     t20,f9.2,t30,'meters',t37,'to',t40,f9.2,t50,'meters (relative'/
     *     t20,'to mean sea level);'//
     *     t5,'Type S, D or K:  ',$)
          ACCEPT 1012, HSCALE
 1012     FORMAT(A1)
          IF (HSCALE .EQ. 'S')  GO TO 1050
          IF (HSCALE .EQ. 'D')  GO TO 1020
          IF (HSCALE .EQ. 'K')  GO TO 1050
C
 1020     TYPE 1021
 1021     FORMAT(//
     *     t5,'Type the bottom height'/
     *     t5,'(in meters, real, relative to mean sea level):  ',$)
          ACCEPT 1022, CBOTTM
 1022     FORMAT(F9.2)
C
          TYPE 1025
 1025     FORMAT(//
     *     t5,'Type the top height'/
     *     t5,'(in meters, real, relative to mean sea level):  ',$)
          ACCEPT 1022, CTOP
C
 1050     CALL ECHO (NO)
          CALL CLR100
          GO TO 1100
```

```
C                       DECIDE WHETHER TO REPLOT
C                       ************************
C
 1100     IF (REPLOT(1)+REPLOT(2)+REPLOT(3)  .EQ.  0)  GO TO 1110
          GO TO 500                          ! replot
C
 1110     ACCEPT 1111, GO                    ! hold plot on screen
 1111     FORMAT(A1)                         ! until user's cue
C
C
C
C                       KEEP EXAMINING THIS DATA FILE?
C                       ******************************
C
 1200     CALL ECHO (YES)
          CALL TYP100
          TYPE 1201
 1201     FORMAT(///T5,'Do you want to keep examining this data file?'/
     *             t5,'(Y or N):  ',$)
          ACCEPT 1202, REPLY
 1202     FORMAT(A1)
          IF (REPLY .EQ. 'Y')  GO TO 1205
          IF (REPLY .EQ. 'N')  GO TO 1300
C
 1205     CALL ECHO (NO)
          CALL CLR100
          BEGRNG = 0.
          ENDRNG = 10000.
          HSCALE = STDRD
          GO TO 500
C
C
C                 SAVE PROCESSED FILE?
C                 ********************
C
 1300     TYPE 1301
 1301     FORMAT(///t5,'Do you want to save the processed file? (Y or N):  ',$)
          ACCEPT 1302, REPLY
 1302     FORMAT(A1)
          IF (REPLY .EQ. 'Y')  GO TO 1305
          IF (REPLY .EQ. 'N')  GO TO 1400
C
 1305     TYPE 1306
 1306     FORMAT(/t5,'Type the name you want the processed file to have;'/
     *           t5,'it should be in DL0 to make sure that there is'/
     *           t5,'enough room, because the file is formatted and'/
     *           t5,'may be large; but be careful NOT to name it'/
     *           t5,'DL0:PROFIL.DAT):  ',$)
          ACCEPT 1307,  PROFIL
 1307     FORMAT(15A1)
C
          TYPE 1308
 1308     FORMAT(//t5,'If necessary, mount the disk which is to contain'/
     *             t5,'the file, then type G, RETURN:  ',$)
          ACCEPT 1309,  GO
 1309     FORMAT(A1)
```

```
      OPEN (UNIT=1, NAME=PROFIL, TYPE='NEW', DISP='KEEP',
     *      FORM='FORMATTED', RECORDSIZE=26, INITIALSIZE=1000)
C
      TYPE 1310
 1310 FORMAT(//t20,'Creating processed file ...'//)
C
      WRITE (1, 1320)  REFALT
 1320 FORMAT(F12.4)
      WRITE (1, 1321)  SPEED
 1321 FORMAT(F8.2)
      WRITE (1, 1322)  RINDEX
 1322 FORMAT(G13.6)
C
      DO 1350  I=1,GOODCT
      II = I
      READ  (2'II)    RANGE, SLHT
      IF (SLHT .EQ. -1.E10)  GO TO 1350
      WRITE (1,1351)  RANGE, SLHT
 1350 CONTINUE
 1351 FORMAT(F14.5,F12.2)
C
      CLOSE (UNIT=1, DISP='KEEP')
C
C
C                    CLOSE RUN
C                    *********
C
 1400 CALL ENDPLT
      CLOSE (UNIT=2, DISP='DELETE')
C
C
C         PLOT ANOTHER PREVIOUSLY PROCESSED/EDITED DATA FILE?
C         ***************************************************
C
      TYPE 1500
 1500 FORMAT(///
     *  t5,'Do you want to plot another previously processed/edited'/
     *  t5,'data file? (Y or N):  ',$)
      ACCEPT 1501, REPLY
 1501 FORMAT(A1)
C
      IF (REPLY .EQ. 'Y')  CALL SETCMD (COMAND)
C
      STOP
      END
```

Auxiliary Routines

```
      PROGRAM SEERAW
C     **************
C
C    This routine is a modification of UNPACK; it allows the user to inspect
C    the raw data files.
C*************************************************************************
C
      BYTE          LDATA(7200), BDATA(14400), INPUT(15), LRE(3), LTE(3),
     *              QR2, LR(2), LS(2), AS(2), LTIME(10), GO
C
      INTEGER*2     TEM, REM, ATTEN, ITEST(25), SHOT, SMIN, SMAX, RANGE,
     *              RPLOT(450), IRE(3), ITE(3), IAS(2),
     *              MONTH, DAY, HOUR, MINUTE, SECOND
C
      EQUIVALENCE   (RANGE,LR(1)), (SHOT, LS(1))
C
      DATA          INPUT/15*'000/
C
C
C        OPEN FILE IN DL0 TO HOLD UNPACKED DATA FILE TO BE INSPECTED
C        **********************************************************
 1100    OPEN (UNIT=2, NAME='DL0:INSPCT.DAT', TYPE='NEW-, DISP='KEEP',
     *         FORM='FORMATTED', RECORDSIZE=24, INITIALSIZE=1000)
C
C
C        QUERY USER FOR NAME OF RAW DATA FILE TO BE INSPECTED
C        ****************************************************
         TYPE 900
  900    FORMAT(//
     *    t5,'Raw data file to be inspected:  ',$)
         ACCEPT 901, INPUT
  901    FORMAT(15A1)
C
C        OPEN RAW DATA FILE
C        ******************
         OPEN (UNIT=1, NAME=INPUT, TYPE='OLD', DISP='KEEP',
     *         FORM='FORMATTED', RECORDSIZE=65)
C
C        READ DATA FROM FILE
C        *******************
C
    5    NB=7200
         NS=450
         NN=32
C
         DO 10 I=1,NS
         LMIN=1+(I-1)*16
         LMAX=LMIN+15
         READ(1,100) (LDATA(L),L=LMIN,LMAX)
   10    CONTINUE
  100    FORMAT(1X,16I4)
C
         CLOSE (UNIT=1, DISP='KEEP')
```

```
C       UNPACK NIBBLES INTO BYTES OF BDATA
C       **********************************
        RANGE=0                                  ! initialize variables
        ATTEN=0
        SHOT=0
        ZAVG=0.
        TAVG=0.
        RAVG=0.
        SZ=0.
        ST=0.
        SR=0.
        NZ=0
        NT=0
        NR=0
        K=1
        SMIN=1
        SMAX=450
C
        DO 30 I=1,NS
C       ************
        ITES=0
        IRES=0
C
        DO 20 J=1,16
        L=J+(I-1)*16
        IF(J.GE.11) GO TO 22
        ITEST(J)=LDATA(L)
        IF(ITEST(J).LT.0) LDATA(L)=LDATA(L)+128
   22   CONTINUE
        BDATA(K)=LDATA(L)-(LDATA(L)/16)*16
        K=K+1
        BDATA(K)=LDATA(L)/16
   20   K=K+1
C
        IF(ITEST(7).LT.0) ITES=80
        IF(ITEST(9).LT.0) IRES=80
        LR(2)=LDATA(L-4)
        LR(1)=LDATA(L-5)
        LS(1)=LDATA(L-1)
        LS(2)=LDATA(L)
        BDATA(K-6)=LDATA(L-2)-(LDATA(L-2)/16)*16
        BDATA(K-5)=LDATA(L-2)/16
        AS(1)=BDATA(K-6)
        AS(2)=BDATA(K-5)
        IAS(1)=AS(1)
        IAS(2)=AS(2)
        LRE(1)=BDATA(K-14)
        LRE(2)=BDATA(K-15)
        LRE(3)=BDATA(K-16)
        LTE(1)=BDATA(K-18)
        LTE(2)=BDATA(K-19)
        LTE(3)=BDATA(K-20)
C
          DO 25 J=1,3
          IRE(J)=LRE(J)
          ITE(J)=LTE(J)
   25     CONTINUE
```

```
      REM=IRE(1)*100+IRE(2)*10+IRE(3)+IRES
      TEM=ITE(1)*100+ITE(2)*10+ITE(3)+ITES
      ATTEN=IAS(1)*10+IAS(2)
      LTIME(1)=BDATA(K-21)
      LTIME(2)=BDATA(K-22)
      LTIME(3)=BDATA(K-23)
      LTIME(4)=BDATA(K-24)
      LTIME(5)=BDATA(K-27)
      LTIME(6)=BDATA(K-28)
      LTIME(7)=BDATA(K-29)
      LTIME(8)=BDATA(K-30)
      LTIME(9)=BDATA(K-31)
      LTIME(10)=BDATA(K-32)
      IF(TEM.NE.0) NT=NT+1
      IF(REM.NE.0) NR=NR+1
      IF(RANGE.NE.0) NZ=NZ+1
      Z=FLOAT(RANGE)
      R=FLOAT(REM)
      T=FLOAT(TEM)
      ZAVG=ZAVG+Z
      TAVG=TAVG+T
      RAVG=RAVG+R
      SZ=SZ+Z**2
      ST=ST+T**2
      SR=SR+R**2
C
      LMIN = 1 + (I-1)*32
      LMAX = LMIN + 31
C
      MONTH = LTIME(1)*10 + LTIME(2)
      DAY   = LTIME(3)*10 + LTIME(4)
      HOUR  = LTIME(5)*10 + LTIME(6)
      MINUTE= LTIME(7)*10 + LTIME(8)
      SECOND= LTIME(9)*10 + LTIME(10)
      WRITE (2,521)   MONTH, DAY, HOUR, MINUTE, SECOND, RANGE
  521 FORMAT(5I3,I9)
C
      IF ((SHOT .LT. SMIN)  .AND.  (SHOT .GT. SMAX))  GO TO 30
C
   30 CONTINUE
C     ********
C
C
C     CLOSE RUN
C     *********
C
      CLOSE (UNIT=2, DISP='KEEP')
C
      STOP
      END
```

```
      PROGRAM CRELST
C     **************
C
C     This routine allows the user to create a file, DY0:FILLST.RAW,
C     to be located on the first raw data disk, which lists the series
C     of raw data files for the flight, along with the disk on which
C     each file resides.
C
C     During execution of the processing/plotting routines, this file
C     is copied to SCR: and read from there, because the first raw data
C     disk may be dismounted during execution.
C*********************************************************************
C
      BYTE       FILNAM(15), REPLY
C
      INTEGER*2  DISKNO
C*********************************************************************
C
      OPEN (UNIT=1, NAME='DY0:FILLST.RAW', TYPE='NEW', DISP='KEEP',
     *       FORM='FORMATTED', RECORDSIZE=16, INITIALSIZE=5)
C
      TYPE 10
   10 FORMAT(//t5,'Type the name of the first raw data file:'/
     *          t5,'(DY0:xxxxxx.xxx, less than 14 characters o.k.) ',$)
      ACCEPT 11, FILNAM
   11 FORMAT(15A1)
      DISKNO = 1
      WRITE (1,12)  FILNAM, DISKNO
   12 FORMAT(15A1,I1)
C
   19 TYPE 20
   20 FORMAT(//t5,'Another raw data file? (Y or N):  ',$)
      ACCEPT 21, REPLY
   21 FORMAT(A1)
      IF (REPLY .EQ. 'Y')  GO TO 29
      IF (REPLY .EQ. 'N')  GO TO 50
C
   29 TYPE 30
   30 FORMAT(//t5,'Type the name of the next raw data file:'/
     *          t5,'(DY0:xxxxxx.xxx, less than 14 characters o.k.) ',$)
      ACCEPT 11, FILNAM
      TYPE 31
   31 FORMAT(t5,'Type the disk number on which this file resides:  ',$)
      ACCEPT 32, DISKNO
   32 FORMAT(I1)
      WRITE (1,12)  FILNAM, DISKNO
      GO TO 19
C
   50 CLOSE (UNIT=1, DISP='KEEP')
      STOP
      END
```

Command Files

The command files execute the various routines. The following list describes them briefly; listings are given on the following pages.

RALTIM.COM	Runs the routines which unpack the raw altimetry data, convert it to horizontal range and ground height above sea level and plot it to show transect of terrain.
RPLOT.COM	Directs runs which only plot previously processed data.
COPY.COM	An auxiliary command file which copies the previously processed/edited data file named by the user (in COPY.COM itself) to DL0 and gives it a standard name, for subsequent use by PLOT. (The technique used here is: RPLOT.COM first calls the system editor to edit COPY.COM; using the editor, the user places the name of the desired data file in COPY.COM; then RPLOT executes COPY.COM, which actually copies the data file to DL0.)
CRELST.COM	Runs CRELST, which creates a file on the first raw data disk listing the raw data files, along with the disk numbers on which they reside. (This file is used by UNPACK.)
SEERAW.COM	Runs SEERAW, which allows the user to inspect raw data files.
SEEAUX	Fortran routine; asks user if another raw data file is to be inspected; if it is, a call is made to the system subroutine SETCMD to rerun SEERAW.COM.

Additionally, there are various command files which handle the editing, compiling and linking of the different routines and several FORTRAN routines which are used by the above command files to display instructions to the user to mount floppy disks on the proper drives.

```
!   RALTIM.COM
!   **********
!
!   This command file directs the processing of the altimetry data,
!   from the unpacking of the raw data through the plotting of the
!   processed data.
!*****************************************************************
!
INIT/NOQ SCR:
INIT/NOQ SD0:
!
!     Instruct the user to have the raw data files to be processed on
!     hand, to load the required floppy disks and to exit from the ed-
!     itor
!     ***************************************************************
!
FORT/OBJ:SD0:ALTIM.OBJ/CODE:THR/EXT/WA/LIST:SD0:ALTIM.LST COM:ALTIM.MS1
LINK/EXE:SD0:ALTIM.SAV SD0:ALTIM.OBJ,SY:FORLIB
RUN SD0:ALTIM.SAV
DELETE/NOQ SD0:(ALTIM.*)
!
!            Display the directory on the first raw data disk
!            ************************************************
!
EDIT/INSPECT DY0:DIRECT.ORY
!
!                 Copy list of raw data files to SCR
!                 **********************************
!
COPY DY0:FILLST.RAW SCR:FILLST.RAW
!
!                     Run the unpacking routine
!                     *************************
RUN DY1:UNPACK.SAV
!
!                     Run the processing routine
!                     **************************
RUN DY1:PROCES.SAV
!
!                     Run the plotting routine
!                     ************************
INIT/NOQ VM:
RUN DY1:PLOT.SAV
```

```
!   RPLOT.COM
!   *********
!
!   This command file directs the execution of a run where a previously
!   created processed file is plotted.
!*****************************************************************
!
INIT/NOQ SCR:
!
!           Tell user to mount processing/plotting disk in DY1
!           ************************************************
!
FORT/OBJ:SCR:ALTIM.OBJ/CODE:THR/EXT/WA COM:ALTIM.MS2
LINK/EXE:SCR:ALTIM.SAV SCR:ALTIM.OBJ,SY:FORLIB
RUN SCR:ALTIM.SAV
DELETE/NOQ SCR:ALTIM.*
!
!
!              Edit the auxiliary command file which copies the
!              previously processed/edited file to a file in DL0
!              with a standard name
!              ************************************************
!
EDIT COM:COPY.COM
!
!
!              Now run the command file that was just edited
!              *********************************************
!
@COM:COPY.COM
!
!
!              Run PLOT
!              ********
INIT/NOQ VM:
RUN DY1:PLOT.SAV
```

```
!   COPY.COM
!   ************
!
!   This command file is edited by the user during execution; it copies
!   a previously processed/edited data file to DL0 and gives it a stan-
!   dard name, for subsequent use by PLOT.
!*****************************************************************
!
!   DIRECTIONS TO USER:
!   *******************
!
!   1. Type the name (including device) of the previously processed
!      data file to be plotted in the spaces indicated below (first
!      erase the name that is there now; the name may be less than 14
!      characters):
!
!      DEV:XXXXXX.XXX
COPY   DL0:FLIT02.P01  DL0:PROFIL.DAT
!
!   2. If the file to be plotted is on a floppy disk, mount that disk
!      in drive DY0 (since the processing/plotting disk is already in
!      DY1).
!
!   3. Exit from the editor (by typing PF1,7 on the keypad, then
!      E X I T, then ENTER on the keypad).
```

```
!   CRELST.COM
!   **********
!
!   This command file runs CRELST, which creates a file on the first
!   raw data disk listing the raw data files, along with the disk
!   numbers on which they reside.
!*****************************************************************
!
! Tell user to mount first raw data disk in DY0
!
FORT/OBJ:SCR:CRELST.OBJ/CODE:THR/EXT/WA COM:CRELST.MS1
LINK/EXE:SCR:CRELST.SV1 SCR:CRELST.OBJ,SY:FORLIB
RUN SCR:CRELST.SV1
!
RUN DY1:CRELST.SAV
DIR/ORDER/FULL DY0:
```

```
!   SEERAW.COM
!   **********
!
!   This command file runs SEERAW, which allows the user to inspect raw
!   data files.
!*****************************************************************
!
INIT/NOQ SCR:
!
!   Tell user to mount raw data disk to be inspected in DY0
!   and processing/plotting disk in DY1
!   ***************************************************
!
FORT/OBJ:SCR:ALTIM.OBJ/CODE:THR/EXT/WA COM:ALTIM.MS3
LINK/EXE:SCR:ALTIM.SAV SCR:ALTIM.OBJ,SY:FORLIB
RUN SCR:ALTIM.SAV
DELETE/NOQ SCR:ALTIM.*
!
!
!   Display directory of the disk in drive DY0
!   ******************************************
!
DIR/ORDER/FULL DY0:
!
!
!   Run SEERAW
!   **********
!
RUN DY1:SEERAW.SAV
!
!
!   Use system editor to inspect the unpacked file
!   **********************************************
!
EDIT/INSPECT DL0:INSPCT.DAT
!
!
!   Delete the file just inspected from DL0
!   ***************************************
!
DELETE/NOQ DL0:INSPCT.DAT
!
!
!   Inspect another raw data file?
!   ******************************
!
FORT/OBJ:SCR:SEEAUX.OBJ/CODE:THR/EXT/WA COM:SEERAW.AUX
LINK/EXE:SCR:SEEAUX.SAV SCR:SEEAUX.OBJ,SY:FORLIB
RUN SCR:SEEAUX.SAV
DELETE/NOQ SCR:SEEAUX.*
```

```
      PROGRAM SEEAUX
C     **************
C
C  This routine asks the user if he wants to examine another raw data
C  file; if he does, a call is made to the system subroutine SETCMD to
C  loop back and re-execute the command file SEERAW.COM.
C
C*********************************************************************
C
      BYTE    COMAND(15), REPLY
C
      DATA    COMAND/'@','C','O','M',':','S','E','E','R','A','W','.',
     *                'C','O','M'/
C
C*********************************************************************
C
      TYPE 10
   10 FORMAT(////
     *  t5,'Do you want to examine another raw data file? (Y or N):
     *  ',$)
      ACCEPT 11, REPLY
   11 FORMAT(A1)
C
      IF (REPLY .EQ. 'Y')  CALL SETCMD (COMAND)
C
      STOP
      END
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-100687		

4. Title and Subtitle	5. Report Date
A Computer Code to Process and Plot Laser Altimetry Data Interactively on a Microcomputer	May 1987
	6. Performing Organization Code: 723.0
7. Author(s): H. G. Safren and J. L. Bufton	8. Performing Organization Report No.: 86B0419
	10. Work Unit No.
9. Performing Organization Name and Address: Goddard Space Flight Center, Greenbelt, Maryland 20771	11. Contract or Grant No.
	13. Type of Report and Period Covered: Technical Memorandum
12. Sponsoring Agency Name and Address: National Aeronautics and Space Administration, Washington, D.C. 20546-0001	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

A computer program, written in FORTRAN, is described which uses a microcomputer to interactively process and plot laser altimetry data taken with a laser altimeter currently under development at the Goddard Space Flight Center, Greenbelt, MD. The program uses a plot routine written for a particular microcomputer, so that the program could only be implemented on a different computer by replacing the plot routine. The altimetry data are taken from an aircraft flying over mountainous terrain. The program unpacks the raw data, processes it into along-track distance and ground height and creates plots of the terrain profile. A zoom capability is provided to expand the plot to show greater detail, along either axis, and provision is made to interactively edit out spurious data points.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Laser altimetry, microcomputer	Unclassified - Unlimited Subject Category 17

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