UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

Geomagnetic Data Utility Programs

for the HP9640A

by

David V. Fitterman

Open-File Report 81-360

1981

This report is preliminary and has not been reviewed for conformity with the U.S. Geological Survey standards. Any use of trade names is for descriptive purposes only and does not imply endorsement by the USGS.

Contents

Figures

Tables

1. Introduction

This report describes a collection of programs used for the selection, manipulation, and display of geomagnetic variation data. Input for these utility programs are 9-track magnetic source tapes created by program TRANZ (D. V. Fitterman, Transcription of geomagnetic variation data from Sea Data cassettes to tape using the HP9640A, USGS Open-File Report No. 81-95, 1981). The data are recorded on magnetic cassettes and transcribed to magnetic tape by program TRANZ.

There are several functions which these programs can perform. The following is a list of the tasks and the programs that performs the tasks.

1. Plot daily magnetograms from source tapes (PLOTØ)
2. Select data segments from source tapes and store in Integer Format disk files (SLECT)
3. Fill in holes in data files (PATCH)
4. Low-pass filter data files (LPBUT)
5. Decimate data files (DECIM)
6. Add or subtract two data files (ADSUB)
7. Multiply or divide the contents of a data file by a constant (MULDV)
8. Remove a linear trend and the mean from a data file (DTRND)
9. List selected portions of a source tape (LSTAP)
10. List selected portions of a disk data file (LSTDS)
11. Plot daily magnetograms from source tape (PLOTØ)
12. Plot the contents of disk files (PLOT3)

The individual chapters of this report contain descriptions of these programs. Input and output file formats for the different programs are described in Appendix A. Appendix B - User´s Guide contains examples of the use of the programs.

Hardware and Software Requirements

The software is intended to be run on a Hewlett-Packard 9460A Multiprogramming System, now superseded by the HP-1000. The hardware consists of a CPU, disk drive, 9-track tape drive, terminal, and a printer/plotter. The plotter used is a Varian Statos 33 and is necessary only for programs PLOTØ and PLOT3.

Most of the software is written in HP FORTRAN IV with a few HP Assembly Language subroutines for special purpose functions. The assembly language routines make use of some HP-21MX instructions which will have to be simulated if the programs are run on the older HP-2100 CPU.

The programs make use of the Spool Monitor Package (SMP or File Manager) to access data files. Consult the HP Batch-Spool Monitor Reference Manual for more information.

The logical unit assignments used in all the programs are shown in Table 1.1.

Table 1.1 Logical unit assignments

LU	Name	Device
1	LUTTY	terminal
6	LUPRT	line printer/plotter
8	LUTAP	magnetic tape

2. SLECT - Data Selection Program

Purpose

Program SLECT is used to extract selected data segments from source tapes. The selected data are stored in disk files. Magnetic and electric field data can be selected by the program. Any missing data (data breaks) can be flagged for correction by other programs.

Description

Before program SLECT is run, a source tape is mounted on the tape drive and positioned at the beginning of the desired file using the File Manager Control (:CN) command. Program SLECT is now run. The program asks if the tape is positioned at the beginning of the file. If the operator does not answer "YE" the program stops. If the tape is in the proper position the header record is read and output. Also output are the day of the year of the RESET TIME and the OFF TIME in the format yyyy.ddd where "yyyy" is the year and "ddd" is the day of the year.

The program now begins processing requests for data selection. The user specifies the time of the first data point wanted by its hour, minute, second, day of the year, and year. The fraction of a year represented by RESET TIME, OFF TIME, and the starting time of the data segment are computed and stored in DAYI, DAYJ, and DAYN respectively. If DAYN does not fall in the closed interval between DAYI and DAYJ, a new starting time is requested.

The number of data points to be selected from the tape is input next. This number does not have to be less than the maximum integer value of 32767 since it is stored as a floating point number. A four character file name is requested, which becomes the prefix of a 6 character file name. No spaces are allowed in the name. The created files have the format "xxxx0c", where "xxxx" is the input file name, and "c" is the data component designator. The designators are X, Y, Z, E, and F for the Hx, Hy, and Hz magnetic, and Ex and Ey electric fields respectively. The "0" in the fifth character of the file name designates that this is "original" data. The user is next asked to indicate the desired data channels to be selected from the tape by typing a "1" if the data are wanted and a "0" if they are not.

The program computes the tick value (2 ticks equals 1 second) of segment starting time (CLKS) and begins searching the tape for a clock value which equals or exceeds CLKS. When this happens the tape record (IREC), subrecord (ISREC), scan number (ISCAN), clock value of the scan (CLK), and CLKS are printed. The output files are created. If an output file can not be created, the file name and error code are printed. The user is then asked for another field name to use. After all of the output files have been created and positioned on the second record, the unpacking loop is entered.

The unpack procedure writes the desired data channels into an Integer Format file. Whenever a data break is detected (indicated by the fourth word of the subrecord being negative) the number of missing data points is computed. The program reports the location of the missing data and how many data points are missing. The user is then asked if SLECT should flag the data break and keep processing data. A response of "YE" will cause values of -1 to be substituted for the missing data. Since the normal range of data values is from 0 to 4095, the data breaks can be easily identified and corrected by other programs (for example program PATCH). When enough data have been extracted, SLECT zeros any unused locations in the last record and outputs it to disk. The header record is updated to reflect the starting time of the data segment and the number of data points it contains. The output files are rewound and the header written in record one. The files are closed and any unused space returned to the system.

The user is asked if any more files are to be created. A reply of "YE" will cause the tape to be repositioned after the header record, and the whole process is started again. Any other response will stop the program and the tape will not be repositioned.

If a data break is encountered, and the user decides not to flag the missing data, data selection is terminated just before the data break. The action of the program is in every other way similar to that of a normal completion except that the output file will be shorter than the requested length.

There are three possible error terminations which can occur during the running of SLECT:

1. An end-of-file mark (EOF) was encountered during the search for the beginning of the desired data segment.
2. There was not enough data available to select a segment of the desired length.
3. An end-of-file mark was encountered during data unpacking.

In the first case the program does not create any files. An error message is printed, and the user is asked if more files are to be created. In the last two cases the output file is created with as much data as was unpacked before the error was encountered. In all three cases, the error message indicates how much data was to be unpacked (FNPT) and how much data was actually unpacked (FIPT). The second and third error conditions are slightly different. The second type of error results from the last tape-file data record not being entirely filled. SLECT actually looks for word four of the subrecord being zero. The search terminates before the read that would have encountered an EOF. Whenever error condition one or three occurs, the tape is backspaced over the EOF. Thus the tape is always positioned between the header record and EOF when SLECT terminates.

Special Requirements

The user should try to pick names for the output files which do not already exist. In the event the output file name chosen is already in use, SLECT will ask the user for another name.

Program Loading

The following commands are used to load program SLECT:

```
:LG,2

:MR,%SLECT

:RU,LOADR,99,6,0,0,2

:SP,SLECT
```

Program Operation

Program SLECT is executed by issuing the following File Manager command:

```
:RU,SLECT
```

The program prompts the user for any required input.

3. PATCH - Data Break Patching Program

Purpose

Program PATCH is used to fill in any flagged data breaks in Integer Format files created by SLECT. The data break is filled in by linear interpolation between the data points on either side of the missing data. This program should be used on data sets with flagged data breaks before any other processing is performed.

Description

PATCH starts by requesting the name of the input file to process. The file must be in Integer Format, but no checking of the file type is done by PATCH. The file is opened, the header record read, and a summary message header written. The summary header indicates the name of the file being processed. If the input file cannot be opened, an error message is printed and the user asked if another file is to be processed.

Next the program searches for a data value of -1 representing a data break. The location of the data break is saved, and the data are now searched for a value not equal to the flag value. This is the end of the data break. If the end of the file is reached before a good data point is found, or if the data break contains more than 1023 points, error messages are printed and the processing terminated. When both ends of the data break have been located, the missing data points are filled in by linear interpolation. A summary of the action taken is printed for each data break encountered. The summary includes the record number, data point number, and data value for the point just before ("FIRST") and just after ("LAST") the data break. The number of interpolated points and the data value change interpolated points ("SLOPE") is printed. The interpolated data values are written in place of the data break flags in the input file. When the end of the file has been reached, the file is closed and the user asked if another file is to be processed.

Special Requirements

Program PATCH should be used on data sets with flagged data breaks before any other processing is done. Since all processing programs treat the flagged data values (-1) as regular data points, the flags could be modified by these programs. If this happens, PATCH would not be able to locate the data breaks.

Program Loading

This program makes use of an assembly language program which must be included during loading. The following loading command sequence can be used.

```
:LG,2

:MR,%PATCH

:MR,%MOVE

:RU,LOADR,99,6,0,0,2

:SP,PATCH
```

Program Operation

Program PATCH is run using the following command

```
:RU,PATCH
```

The program prompts the user for any needed information.

4. LPBUT - Low-Pass Filtering Program

Purpose

Program LPBUT is used to low-pass filter data. This is usually done before a data sequence is decimated to prevent aliasing. An analog Butterworth filter is designed which meets the specified design criteria, and is converted to a digital filter by means of the bilinear transformation. A recursive realization is used for the filter, and it is applied in the forward and reverse directions so that the filter introduces no phase shift. (Refer to Digital Signal Processing, A. V. Oppenheim and R. W. Schafer, p. 195-283, Prentice-Hall, Englewood Cliffs, New Jersey, 1975 for more details and definition of filter design parameters.)

Description

Program LPBUT requests the name of the input file to be processed. The file must be in Integer Format and have the letters "O" or "D" as the fifth character of the name. After a satisfactory name has been input the file is opened. If the name does not satisfy the above criterion or the file cannot be opened, an error message is printed and the user asked if another file should be filtered.

Once the input file has been opened, the file header is read, and the Nyquist frequency of the data set computed. The user then supplies the 3 dB attenuation frequency, the stop frequency, and the attenuation at the stop frequency. Subroutine DEBUT is now executed to do the filter design. This routine designs an analog Butterworth filter, which if cascaded with itself gives the desired frequency response. This is done because the data are filtered in the forward and reverse directions to introduce no phase shift. The analog design frequencies are warped by means of the bilinear transformation for use in determining the analog filter (S-plane) poles. If requested, DEBUT will print the filter design parameters, the S-plane poles, and the Z-transform coefficients.

The design parameters include:

1. WC - the filter cutoff frequency in radians

2. WS - the filter stop frequency in radians

3. DB - the filter attenuation in dB at the stop frequency

4. OWC - The warped digital angular cutoff frequency. (The Nyquist frequency corresponds to π. All other frequencies are fractions of the Nyquist frequency.)

5. OWS - the warped digital angular stop frequency

6. ALPHA - $\log_{10}(\sqrt{2} - 1) = -0.3827757$

7. BETA - $\log_{10}$ (10**DB/20 - 1)

8. AN - floating point order of the filter

9. NORDR - actual integer order filter used in the computations

10. ANC - logarithm of the analog cutoff frequency when the filter order is set to NORDR

11. OWP - 10**ANC. This corresponds to the analog cutoff frequency for one direction of filtering.

12. IODD - set to 1 if NORDR is odd, meaning there is a pole on the negative real axis in the S-plane; otherwise equals 0.

If NORDR is greater than 100, the filter will exceed the storage limits of the program. When this happens, the user is asked to input new filter parameters. Reducing the sharpness of the filter will overcome this problem.

DEBUT next computes the location of the filter poles in the left-hand side of the S-plane. Only the values of the poles in the second quadrant are printed since the poles occur as conjugate pairs. The filter is implemented as a Z-transform product of terms of the form

$$\frac{1 + cz^{-1} + dz^{-2}}{1 + az^{-1} + bz^{-2}}$$

and a gain factor (PROD), which gives the filter unity gain at zero frequency. The coefficients a, b, c, and d are printed if requested. A tabulation of the filter response will be printed if requested.

Once the filter is designed, LPBUT creates an output file named "xxxxLc" from the input file name of "xxxx0c" or "xxxxDc". (See Appendix A for details of file naming conventions.) If a file with this name cannot be created, an error message is printed, and the user asked if another file is to be filtered.

The actual filtering is now done by subroutine FILTR. The filters are initialized to minimize transient effects. After the data have been filtered in both directions, the output file is closed. The user is then asked if another file is to be filtered.

Special Requirements

Program LPBUT requires that the input file name have the format "xxxx0c" or "xxxxDc". It tries to create an output file named "xxxxLc". If it cannot create this file, the input file is not filtered, and the user is asked if another file should be filtered.

Program Loading

The following sequence of commands is used to load program LPBUT:

```
:LG,2

:MR,%LPBUT

:RU,LOADR,99,6,0,0,2

:SP,LPBUT
```

Program Operation

The program is run by issuing the command

```
:RU,LPBUT
```

The program prompts the user for any needed information. While the data are being filtered, the user can keep track of the progress of the operation by using the following system command

```
*BR,LPBUT
```

This command causes program LPBUT to write a message indicating the direction it is filtering the data (IFLAG equals 1 for forward and -1 for reverse filtering), the number of points already filtered on this pass through the data (GDATA), and the number of points in the data sequence (FDATA). The data are filtered in the forward direction first. Filtering takes about 0.0058 seconds/filter-pole/data-point where the number of filter poles is NORDR/2 + IODD. Thus a 5000-point data sequence using a 10-pole filter would take 290 seconds to process.

5. DECIM - Decimation Program

Purpose

Program DECIM is used to decimate or extract parts of an Integer Format file. Data should be low-pass filtered before decimation to prevent aliasing.

Description

Once DECIM is running it asks the user for the name of the input file to be decimated. The file name must have an "O" or "L" in the fifth character position. After the name is input the file is opened. If the file name is incorrect or the file can not be opened, the user is asked if another file is to be decimated. The header record is read, and the number of data points in the file is printed. The user specifies the number of data points to be skipped at the beginning of the input file (NSKIP) and the decimation number (NDEC). If the decimation number is set to n, every n-th data point will be placed in the output file. Setting the decimation number to unity results in data extraction without decimation. The program then reports the maximum number of points which can be put into the output file based on the previous two inputs. The user then indicates how many points are to be put into the output file. If this number is equal to zero, the user is asked for a different skip and decimation number. Inputting a value of -1 terminates processing without writing an output file, and asks the user if another file is to be processed. Once the length of the ouput file has been determined, one output file is created. This file will have the same name as the input file, but with the fifth character in the file name changed to a "D". If the file cannot be created, an error message is printed and the user asked if another file is to be processed.

When a satisfactory output file has been created, the header values are changed to reflect the new effective sample interval and the header is written. The input file data records are now read and every NDEC point is written to the output file. After processing is complete the files are closed and the user asked if another file is to be processed.

Special Requirements

The input file used by program DECIM must have a name of the form "xxxx0c" or "xxxxLc", and an output file name of the form "xxxxDc". If these conditions are not met, an error message will result and processing will be terminated.

Program Loading

DECIM can be loaded with the following File Manager command sequence:

```
:LG,2

:MR,%DECIM

:RU,LOADR,99,6,0,0,2

:SP,DECIM
```

Program Operation

This program is executed by using the following command

```
:RU,DECIM
```

The user is prompted for any additional input which is required.

6. ADSUB - Addition/Subtraction Program

Purpose

Program ADSUB is used to add or subtract the contents of two files and put the results in another file. The input files may be Integer, Real, or Complex Format.

Description

The first information which must be supplied to program ADSUB is the type of files being processed (integer, real, or complex), the operation to be performed (addition or subtraction), and the names of the input files. If either input file cannot be opened, any files which have been opened are closed and the user asked if other files are to be processed.

Once the input files are opened, the header records are read and checked for the discrepancies listed below:

1. Different number of data points in each file.
2. Different effective sample interval
3. Different starting time of the data segments.

The first two conditions result in errors which stop processing, while the last condition generates a warning message and the user is asked if processing should continue.

If everything is satisfactory at this point, the length of the output file is computed, the user asked for the name of the output file, and the output file is created. If an error occurs during file creation, an error message is written, and the user asked if any other files are to be processed. The actual arithmetic is now performed and the results written to the ouput file.

During integer addition and subtraction operations, the maximum deviation from 2048 of the number of counts in the result file is determined. If this number exceeds 2048, the instrument gains in the header record and the integer data values are adjusted to keep all data values in the range of 0-4095. This is done by increasing the magnetometer gain and decreasing the telluric amplifier gain magnitudes. The integer counts in the file are reduced appropriately.

After the calculations are completed, the user is asked if more data are to be processed. Responding "NO" terminates program ADSUB, while a response of "YE" starts the input sequence over again.

Program Loading

Program ADSUB is loaded with the following sequence of commands:

```
:LG,2
:MR,%ADSUB
:RU,LOADR,99,6,0,0,2
:SP,LOADR
```

Program Operation

This program prompts the user for any needed information. It is started running by giving the following command

```
:RU,ADSUB
```

Augent, Minuend, and All That Stuff

As I have difficulty remembering the names of the various components of the arithmetic operations of addition and subtraction, I have included them below. They might be of help to other users when specifying the input files for program ADSUB.

$$\begin{array}{l} \quad\text{Augend} \\ \underline{+\ \text{Addend}} \\ \quad\text{Sum} \end{array} \qquad\qquad \begin{array}{l} \quad\text{Minued} \\ \underline{-\ \text{Subtrahend}} \\ \quad\text{Difference} \end{array}$$

7. MULDV - Constant Multiplication Program

Purpose

Program MULDV is used to multiply the contents of a disk file by a constant. This function might be used to correct for improper gain settings of recorders.

Description

The user first specifies the type of file (integer, real, or complex) to be multiplied and its name. If the input file cannot be opened, the user is asked if another file is to be processed. If the input file can be opened, the user then supplies the factor which the data are to be multiplied by. In the case of real or complex format files, the data are multiplied by the specified constant.

Multiplication of integer format data can result in numbers which exceed the normal range (0-4095) of data values. Before multiplication is done, the data are searched to determine the maximum deviation from 2048, the count value corresponding to zero. If multiplication by the input factor will cause this deviation to exceed 2048, the gains in the header record and the multiplier are scaled so that the maximum deviation after multiplication by the new factor is 2048. After the data have been multiplied, the user is asked if any more data are to be processed.

Special Requirements

Multiplying an integer file by a constant whose absolute value is less than unity will decrease the dynamic range of the data. The user should also be aware that the gains are stored as integer constants, and that inaccuracies will be introduced if the integer arithmetic is not exact, *e.g.*, division of an odd number by 2.

Program Loading

MULDV is loaded with the following commands:

```
:LG,2

:MR,%MULDV

:RU,LOADR,99,6,0,0,2

:SP,MULDV
```

Program Operation

The File Manager command

```
:RU,MULDV
```

is used to run this program. The program prompts the user for any required inputs.

8. DTRND - Trend Removal Program

Purpose

Program DTRND is used to remove the mean or the linear trend from a data set. These functions can be performed independently or simultaneously.

Description

The user specifies an integer file with the letter "O", "L", or "D" as the fifth character of the name (see Appendix A). If the file can be opened the header is read, and the length of the file checked. Files with more than 32767 data points cannot be processed by this program. The data values are read and their sum computed. The first and last data values are stored for use in the computations. The user can specify the following type of terms to remove from the data: (1) straight line between end points of data set (SLOPE), (2) the average value of the data set (DC), or (3) both of the previous terms (BOTH). The following equation is used to form the new count value (C′) from the old value (C).

$$C'_i = C_i - i * \mathrm{SLOPE} + \mathrm{BIAS} \qquad i=i,\ldots, N$$

where i is the data point number. The values of SLOPE and BIAS for the three types of trend removal are given in the table below.

Table 8.1 Values of SLOPE and BIAS used in trend removal

Trend Removal Type	SLOPE	BIAS
linear straight line	$\frac{C_N - C_1}{N-1}$	0.0
average value	0.0	$2048 - \sum_i C_i/N$
both	$\frac{C_N - C_1}{N-1}$	$2048 - \frac{C_N - C_1}{2} - \sum_i C_i/N$

After the trend removal is completed, the file is closed and a summary is printed on the line printer. The user is then asked if any other files are to be processed.

Special Requirements

Program DTRND requires that the input file have either the letter "O", "L", or "D" as the fifth character of its name. The file cannot contain more than 32767 points.

Program Loading

Program DTRND is loaded using the following command sequence:

```
:LG,2
:MR,%DTRND
:RU,LOADR,99,6,0,0,2
:SP,DTRND
```

Program Operation

Program DTRND is run using the following command

```
:RU,DTRND
```

The user is prompted for any required input.

9. LSTAP - Source Tape Listing Program

Purpose

This program is used to list selected portions of source tapes.

Description

The tape to be listed is mounted on the tape drive, and File Manager commands are used to position the tape at the beginning of the desired file. The program begins by asking if the tape is positioned at the beginning of the file. Any response other than "YE" will cause the program to stop. If the program is properly positioned, the header record is read and printed. The program is then given the first (START) and last (STOP) record and subrecord number to be listed. The program checks that the STOP value comes after the start value, and the values given are within allowable limits. The user also specifies the output format, either HEADER or HEADER + DATA. The former choice prints only the information contained in the subrecord header (subheader), while the latter chosen also prints the data values. The data values are written in counts.

After the input information has been supplied, the tape is positioned at the proper record. This function is performed by subroutine LOCAT. The current record position of the tape is maintained by the program so that searching can be done in a forward or reverse direction. Once the proper record is located the data display loop is entered. The appropriate subrecords are printed in the specified format by subroutine OUT. If the output request spans more than one tape record, additional records are read and output until all of the desired data segment has been displayed.

If an EOF is encountered during the initial search or while reading subsequent records for display, a message is written. The tape is then repositioned after the header record, and the user asked if another data segment is to be listed. When no more data segments are to be listed, the tape is positioned before the header record and the program stops.

Special Requirements

Tape files must be in a Source File format. Any tape that can be read by program SLECT can also be read by program LSTAP.

Program Loading

Use the following command sequence to load program LSTAP

```
:LG,2

:MR,%LSTAP

:RU,LOADR,99,6,0,0,2

SP,LSTAP
```

Program Operation

Program LSTAP is run using the command

```
:RU,LSTAP
```

The program prompts the user for any needed input.

10. LSDSK - Disk File Listing Program

Purpose

Program LSDSK is used to list the contents of Integer Format disk files such as those created by programs SLECT, LPBUT, or DECIM.

Description

The user supplies the name of the file to be listed. The file is opened, and the number of records in the file is displayed. The user is then asked if the entire file is to be printed. A response of "YE" results in each record of the file being listed. Any other response allows the user to select individual records to be listed. Supplying a non-positive record number terminates record listing for this file. The user is then asked if another file is to be listed.

Special Requirements

The files read by program LSDSK are opened as File Manager type 1 files. This is the file type used for Integer Format files. Use of the program with other file types may give unsatisfactory results. All records, including the header record, are printed as integers. This may make the ASCII and floating point format information difficult to interpret.

Program Loading

The following procedure is used to load program LSDSK:

```
:LG,2

:MR,%LSDSK

:RU,LOADR,99,6,0,0,2

:SP,LOADR
```

Program Operation

The program is run using the command

```
:RU,LSDSK
```

The program prompts the user for any needed information.

11. PLOTØ - Daily Magnetogram Plotting Program

Purpose

Program PLOTØ is used to plot daily magnetograms from geomagnetic source tapes produced by program SLECT. By using the Varian printer/plotter in strip-chart mode the plots can be made quite quickly. The disadvantage of this method is that only minimal documentation can be put on the plots.

Description

The tape to be plotted is mounted and positioned to the beginning of the first file to be plotted. PLOTØ is started, and the header record is read and written on the terminal. The minimum and maximum values for each data channel are printed, and the user types in the plotting limits for each channel. The different plotting marks are computed, and the plotting of the daily magnetograms now begins.

The daily magnetograms are 18" long (1"=1 hour) and 2" wide per channel. Each day contains 1801 plot lines corresponding to a different time of day. One of three masks corresponding to the line being plotted is loaded into the output buffer. The three masks correspond to the first and last line of a day, hour mark lines, and all other lines in a day. The data since the previous plot line up to and including the time of the present plot line is logically OR-ed with the mask in the output buffer, and then plotted. When the last line of the day has been plotted, the paper is advanced to make a margin, and the plotting procedure started for the next day.

When no more data exists on the input file, the rest of the masks needed to complete the day are plotted. The user is then asked if the next file should be plotted.

Special Requirements

The input tape used by program PLOTØ must be in Source Tape Format. A Varian Statos printer/plotter is required to do the raster-mode plotting.

Program Loading

The following sequence of commands is used to load this program:

```
:LG,2
:MR,%PLOTØ
:MR,%INDOT
:RU,LOADR,9,9,6,0,0,2
:SP,PLOTØ
```

Subroutine INDOT is an assembly language routine that inserts dots in the output raster buffer.

Program Operation

This program prompts the user for any needed input. The program is started running with the command

```
:RU,PLOTØ
```

12. PLOT3 - Disk File Plotting Program

Purpose

Program PLOT3 is used to produce plots of data sets stored in Integer Format disk files. This includes files produced by programs SLECT, LPBUT, and DECIM.

Description

The user supplies the name of an Integer Format disk file to be plotted. The file is opened, and the header record is read. If the file contains more than 32,767 data points, a message is printed, the file close and processing terminated. If the file is plottable, some of the header record information is printed for use by the operator. Subroutine CHANL is called to determine the data channel being plotted. This is accomplished by examining the sixth letter of the file name. If the sixth character in the file name is not "X", "Y", "Z", "E", or "F", the user is asked to indicate the data component. This information is needed for computation of the gain factor for the data.

The user now specifies the size and tick mark interval of the plot. The program can be asked to search for the minimum and maximum data values so the user will have some guide as to what values to assign FMAX and FMIN. Character strings for the title and subtitle that will appear above the plot can also be input. If these are to be left blank, the input should be at least one space. The last input the user gives is whether or not more than one copy of the plot is wanted. Subroutines BOX and DATA are now called to plot the annotated axes and the data points, respectively.

As the plot vectors are generated, they are sorted in blocks of 64 and written into a file named "VECTRS". After the last vectors have been written, this file is closed and Program MERGE is scheduled. This program merges the blocks of sorted vectors. When it is done, program PLOT is scheduled by program PLOT3. Program PLOT does the vector-to-raster conversion and the actual plotting of the data. When plotting is finished, program PLOT3 asks if another file is to be plotted.

Special Requirements

The input file plotted by program PLOT3 must be in Integer Format and contain 32,767 or fewer data points. An output file named "VECTRS" is created for storing the plotting vectors. If this file exists when program PLOT3 is run an error will result. After abnormal termination of PLOT3, this file should be purged (:PU,VECTRS) to prevent this problem.

PLOT3 schedules programs MERGE and PLOT. They should have temporary ID segments assigned to them before PLOT3 is run to prevent the occurrence of SC05 errors by issuing the following commands:

```
:RP,MERGE

:RP,PLOT
```

Program Loading

Program PLOT3 is loaded using the following commands:

```
:LG,2

:MR,%PLOT3

:RU,LOADR,99,6,0,0,2

:SP,PLOT3
```

Program Operation

This program prompts the user for all required inputs. The program is started operating with the command:

```
:RU,PLOT3
```

13. MERGE - Plot Vector Merging Program

Purpose

This program is scheduled by program PLOT3 to merge the sorted plot vectors.

Description

This program was not written by the author of this report, nor is the author familiar with the details of its operation. It has been used by the author as a black-box, and has been included in this report for the sake of completeness. Section 17 contains a listing of this program.

Special Requirements

The program creates a temporary scratch file named "MERGER", which is used during the merging of the plot vectors.

Program Loading

The program is loaded using the following commands.

```
:LG,2
:MR,%MERGE
:RU,LOADR,99,6,0,0,2
:SP,MERGE
```

Program Operation

This program is usually scheduled by program PLOT3, which also passes several parameters to program MERGE. The calling sequence if initiated by an operator would be

```
:RU,MERGE,2Hfi,2Hln,2Ham,-10,0
```

where "filnam" represents the name of the file containing the vectors to be merged ("VECTRS"). This program requires no operator intervention.

14. PLOT - Vector Rasterization Program

Purpose

Program PLOT is used to rasterize the sorted vector file, and plot the raster.

Description

Program PLOT is provided with the name of the file containing the sorted vectors to be rasterized and plotted. The vector file is opened, and the assembly language subroutine VRAS which does the rasterizing and plotting is called. After plotting, a check is made to see if more than one copy of the plot might be wanted. If this is the case, the user is asked if another copy should be printed. An affirmative response will cause the program to plot another copy of the file. Any other response will cause the vector file to be purged and the program halts.

Special Requirements

This program can only rasterize plots with a finite number of active vectors. An active vector is one which intersects the horizontal line currently being plotted. The maximum number of active vectors is equal to the dimension of array IRBUF divided by 6. If a rasterizing error occurs it is caused by the number of active vectors exceeding this limit. The remedy is to increase the dimension of array IRBUF, or reduce the number of lines in the plot. With the present system configuration, increasing the size of the array is not possible.

Program Loading

This program is loaded using the following commands:

```
:LG,2

:MR,%PLOT

:MR,%VRAS
```

```
:RU,LOADR,99,6,0,0,2

:SP,PLOT
```

Program Operation

This program is usually scheduled by program PLOT3, which also passes several parameters to program PLOT. The calling sequence if initiated by an operator would be:

```
:RU,PLOT,2Hfi,2Hln,2Ham,-10,IRPL
```

where "filnam" is the vector file name, and IRPL is set to -1 to make more than one copy of the plot. A value of zero for IRPL will produce just one copy of the plot. The program prompts the user for any additional input.

15. Appendix A -- Data Formats and File Names

Data Formats

There are several data formats mentioned in this report that are described below. Most users will be concerned with Source-Tape and Integer formats. In fact, only programs ADSUB and MULDV will correctly perform operations on Real and Complex format files. Since none of the programs described in this report can create Real or Complex format files, the user need not be concerned about them. They have been mentioned only for the sake of completeness and possible expansion.

Source-Tape Format

This is the format of tapes that are read by programs SLECT and LSTAP. A complete description of this format is given in USGS Open-File Report 81-95, Appendix A (D. V. Fitterman, Transcription of geomagnetic variation data from Sea Data cassettes to tape using the HP9640A, USGS Open-File Report 81-95, 1981).

Integer Format

Integer Format files are created by program SLECT. Files with this format serve as input to all of the programs described in this report. The files consist of a 128-word header record followed by 128-word data records. The words are 16 bits long. The values of the data words should lie in the range of 0 to 4095. The units of the data are counts. Any unused data words at the end of the last record are set to zero.

The header record has essentially the same format as Source-Tape Files for the first 60 words. The header record format is described in Table 15.1. Some of the parameters are not used by any of the parameters described in this report, but have been included for completeness.

Table 15.1 Integer Format file header record format

Word	Contents
1	Transcription version number
2	Day of year of transcription
3	Year of transcription
4	Tape file number (0-32767)
5	1st and 2nd character of location code (ASCII)
6	3rd and 4th character of location code (ASCII)
7	Cassette ID number (0-99)
8	Instrument number (1-31)
9	Scanrate (0-7), NRATE (Original sample interval = 2**(NRATE-1) seconds)
10	Channels per scan (1-7), NCHAN
11	Clock reset time, hours
12	Clock reset time, minutes
13	Clock reset time, day
14	Clock reset time, month
15	Clock reset time, year
16	Clock off time, hour
17	Clock off time, minute
18	Clock off time, day
19	Clock off time, month
20	Clock off time, year
21	Stop watch time, minute
22	Stop watch time, second
23	Stop watch time, tenths of second
24	Number of words per cassette record
25	Number of cassette records per disc record (always 32)
26	Number of words per cassette record
27-51	Comment field (50 ASCII characters)
52	Number of words per subrecord, NWORD
53	Number of scans per subrecord, NSCAN (NSCAN = integer (24/NCHAN)
54	Hx gain in nT/2048 counts (Value of 0 indicates a default value of 1000 nT/2048 counts.)

Word	Contents
55	Hy gain
56	Hz gain
57	Ex gain, >0 north end (+), <0 south end (+)
58	Ey gain, >0 east end (+), <0 west end (+)
59	Ex line length in meters
60	Ey line length in meters
61	NHOUR (Starting time of data segment)
62	NMIN (Starting time of data segment)
63	NSEC (Starting time of data segment)
64	NDAY (Starting time of data segment)
65	NYEAR (Starting time of data segment)
66	Number of data points in data segment, NPT (0-32767) Set to -1 when greater than 32767. Then use FNPT in word 127 and 128.
67	Decimation number, NDEC. Equals 1 for no decimation
68	Original sample interval in ticks 1 tick = 1/2 second)
69-71	Reserved
72-126	Not used
127-128	Number of data points in floating point format.

Real Format

Real Format files have the same header as Integer Format files. The data records, however, contain only 64 real floating point data values. These data values have been converted from offset binary integer counts to true data values with units of nanoteslas or mV/km.

Complex Format

Complex Format files have the same header information as Integer Format files. There are, however, only 32 complex data values per data record.

File Names

Some of the programs described in this report use and require the use of the file naming conventions desribed below. File names can contain up to six characters. The standard name is of the form

xxxxtc

where "xxxx" is a four character location identifier, "t" is the file type designator, and "c" is the data component (or channel) designator. Only three file type designators are used by the programs described in this report; they are: (1) "O" - original data, (2) "L" - low-pass filtered data, and (3) "D" - decimated data. Table 15.2 lists the allowable input and output types for all utility programs.

Table 15.2 Allowable input and output file type designators for utility programs

Program	Input	Output	Note
SLECT	S	0	1
PATCH	0, L, D	-	2
LPBUT	0, D	L	
DECIM	0, L	D	
ADSUB	0, L, D	-	3
MULDV	0, L, D	-	3
DTRND	0, L, D	-	2
LSTAP	S	-	
LSDSK	0, L, D	-	4
PLOTØ	S	-	
PLOT3	0, L, D	-	4

File Type Designators

S = source tape

0 = original data

L = low-pass filtered data

D = decimated data

Notes

1 - Tries to create output type indicated, but allows user to override if not possible.

2 - Writes output to input file.

3 - Allows input and output files with any name, format can be Integer, Real, or Complex.

4 - Allows input file with any type designator, but format must be Integer.

The component designations are shown in Table 15.3

Table 15.3 File component designators

Designator	Data Component
X	magnetic field Hx
Y	magnetic field Hy
Z	magnetic field Hz
E	electric field Ex
F	electric field Ey

16. Appendix B -- User's Guide

This section shows examples of the operation of the utility programs described in this report, and gives an explanation of the resulting output. It is not intended to be an exhaustive description of the functioning of the programs. For a more detailed description see the Program Description sections or look through the program listings.

This section contains figures that are copies of actual terminal sessions and the associated output. The figures are keyed with circled numbers which correspond to the comments in the text.

PLOTØ

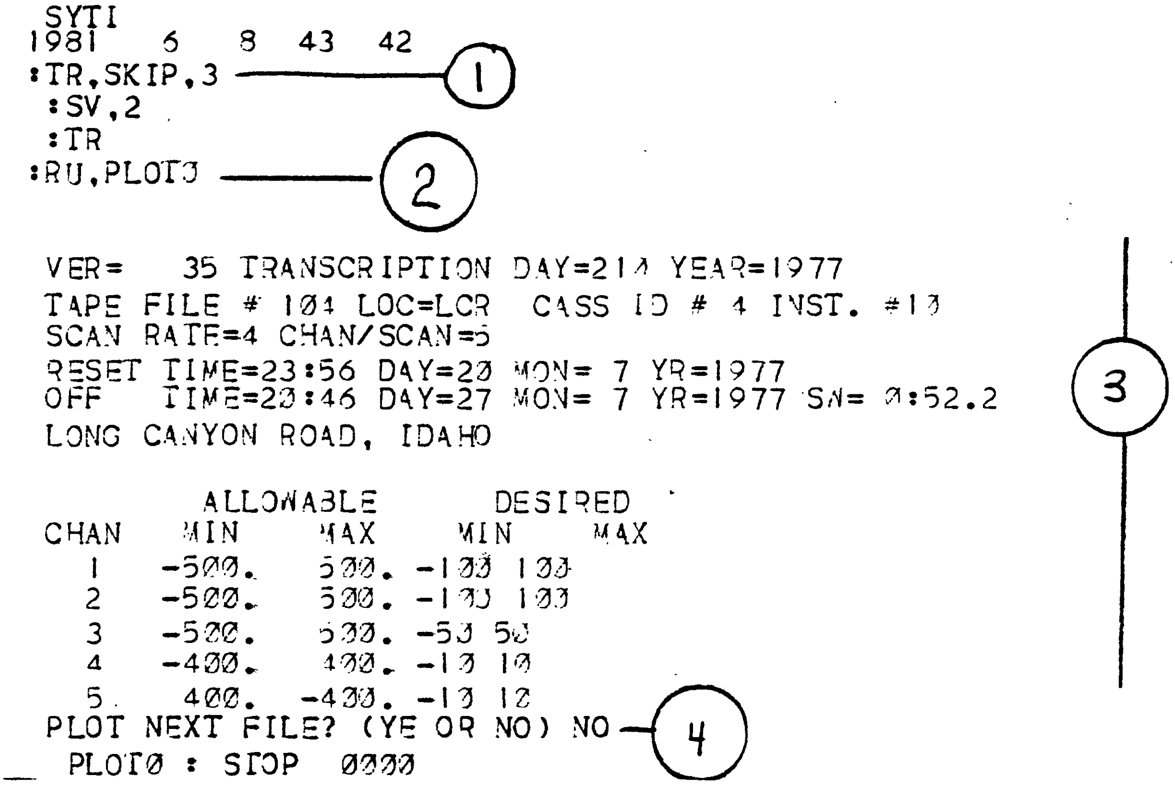
Refer to Figure 16.1 for the following discussion of the operation of program PLOTØ.

1. A source tape created by the geomagnetic transcription software is mounted and this command is given to skip forward three files.
2. Program PLOTØ is run to plot daily magnetograms.
3. The header information is printed followed by the full-scale data values. The user inputs the values under the columns labeled "DESIRED".
4. When the plotting is finished, the user responds with "NO" to prevent the plotting of the next file.

A portion of one of the daily magnetograms is shown in Figure 16.2. The plot size has been reduced.

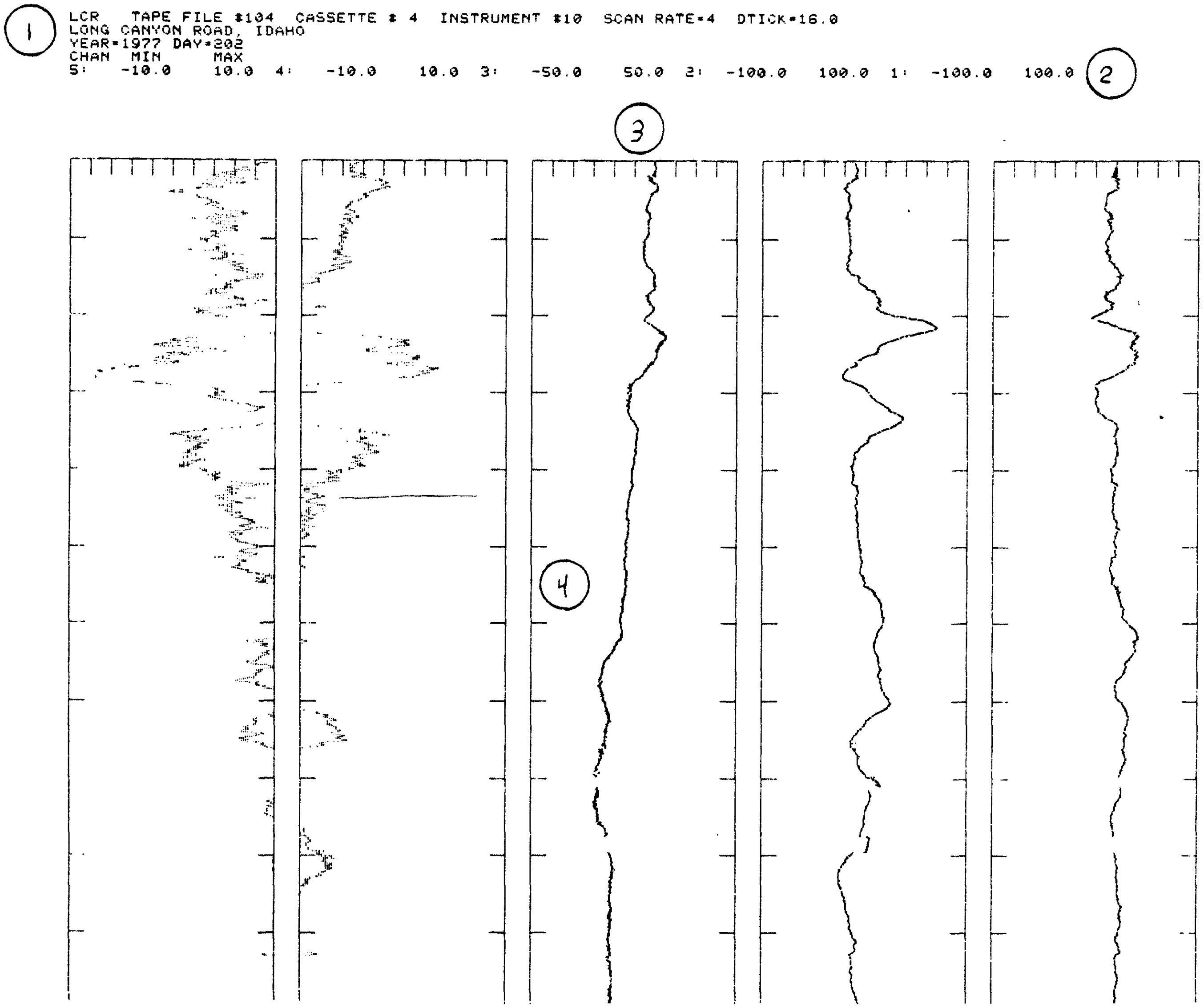
1. Header information. The sample interval in ticks (1/2 seconds) is given by DTICK. Also printed are the year and day number of the plot.
2. This row of numbers gives the minimum and maximum values for the plots. Channel 1 is to the right and corresponds to the X-component of magnetic field.

Figure 16.1 Example of running program PLOTØ



```
 SYTI
1981   6   8  43  42
:TR,SKIP,3
 :SV,2
 :TR
:RU,PLOTØ

 VER=   35 TRANSCRIPTION DAY=214 YEAR=1977
 TAPE FILE # 1Ø4 LOC=LCR  CASS ID # 4 INST. #1Ø
 SCAN RATE=4 CHAN/SCAN=5
 RESET TIME=23:56 DAY=2Ø MON= 7 YR=1977
 OFF   TIME=2Ø:46 DAY=27 MON= 7 YR=1977 SN= Ø:52.2
 LONG CANYON ROAD, IDAHO

          ALLOWABLE       DESIRED
 CHAN   MIN     MAX     MIN    MAX
   1   -5ØØ.   5ØØ. -1ØØ 1ØØ
   2   -5ØØ.   5ØØ. -1ØØ 1ØØ
   3   -5ØØ.   5ØØ. -5Ø 5Ø
   4   -4ØØ.   4ØØ. -1Ø 1Ø
   5.   4ØØ.  -4ØØ. -1Ø 1Ø
 PLOT NEXT FILE? (YE OR NO) NO
  PLOTØ : STOP  ØØØØ
```

Figure 16.2 Daily magnetogram plotted by program PLOTØ



```
LCR   TAPE FILE #104  CASSETTE # 4  INSTRUMENT #10  SCAN RATE=4  DTICK=16.0
LONG CANYON ROAD, IDAHO
YEAR=1977 DAY=202
CHAN  MIN      MAX
5:   -10.0     10.0  4:   -10.0     10.0  3:   -50.0     50.0  2:  -100.0    100.0  1:  -100.0    100.0
```

3. Each plot scale is divided into 10 divisions. This plot, channel 3, goes from -50 nT to 50 nT and is the Z magnetic field component. Each scale division represents one-tenth of the full-scale range or 10 nT in this example.

4. The time axis has tick marks every hour. On the full-size plots there is 0.75" between tick marks. Each plot is 24 hours long and starts at midnight.

SLECT

After plotting the magnetogram, the next operation is to extract data segments for analysis. This is done using program SLECT. (See Figure 16.3.)

1. SLECT is run, and the user is asked if the source tape is positioned at the beginning.

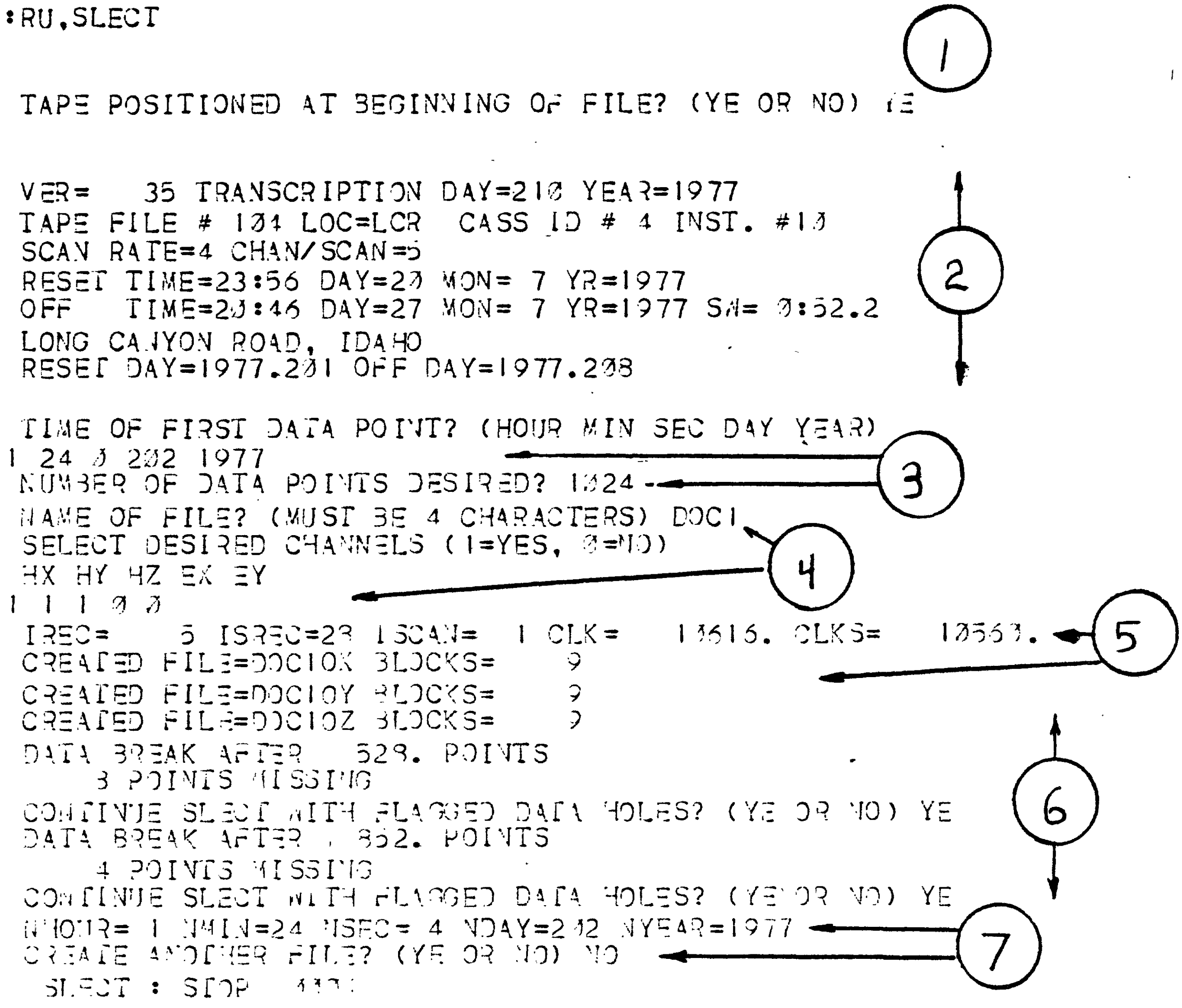
2. The header record is written, and printed. Notice also that the RESET and OFF days are also printed in the form yyyy.ddd where the integer part is the year and the fractional part is the day number.

3. The user requests a starting time of 01:24:00 on day 202 of year 1977 for the data segment to be extracted. A total of 1024 data points are to be selected for each file created.

4. The user specifies a four character prefix (DOC1) for the files which will be created. The user indicates that only the magnetic data channels (HX, HY, and HZ) are to be extracted. At this point the program starts searching the tape for the requested data.

5. The desired data are found in tape record 5, subrecord 28, scan 1. The clock value in this subrecord was 10616, and the clock value corresponding to the requested first data point time was 10560. SLECT has created three files, each nine blocks long to put the data into.

Figure 16.3 Example of running program SLECT.



```
:RU,SLECT

 TAPE POSITIONED AT BEGINNING OF FILE? (YE OR NO) YE

 VER=   35 TRANSCRIPTION DAY=21Ø YEAR=1977
 TAPE FILE # 1Ø4 LOC=LCR  CASS ID # 4 INST. #1Ø
 SCAN RATE=4 CHAN/SCAN=5
 RESET TIME=23:56 DAY=2Ø MON= 7 YR=1977
 OFF   TIME=2Ø:46 DAY=27 MON= 7 YR=1977 SW= Ø:52.2
 LONG CANYON ROAD, IDAHO
 RESET DAY=1977.2Ø1 OFF DAY=1977.2Ø8

 TIME OF FIRST DATA POINT? (HOUR MIN SEC DAY YEAR)
1 24 Ø 2Ø2 1977
 NUMBER OF DATA POINTS DESIRED? 1Ø24
 NAME OF FILE? (MUST BE 4 CHARACTERS) DOC1
 SELECT DESIRED CHANNELS (1=YES, Ø=NO)
 HX HY HZ EX EY
1 1 1 Ø Ø
 IREC=     5 ISREC=28 ISCAN=  1 CLK=    13616. CLKS=    1Ø563.
 CREATED FILE=DOC1OX BLOCKS=     9
 CREATED FILE=DOC1OY BLOCKS=     9
 CREATED FILE=DOC1OZ BLOCKS=     9
 DATA BREAK AFTER   528. POINTS
     3 POINTS MISSING
 CONTINUE SLECT WITH FLAGGED DATA HOLES? (YE OR NO) YE
 DATA BREAK AFTER   852. POINTS
     4 POINTS MISSING
 CONTINUE SLECT WITH FLAGGED DATA HOLES? (YE OR NO) YE
 NHOUR= 1 NMIN=24 NSEC= 4 NDAY=2Ø2 NYEAR=1977
 CREATE ANOTHER FILE? (YE OR NO) NO
  SLECT : STOP  [illegible]
```

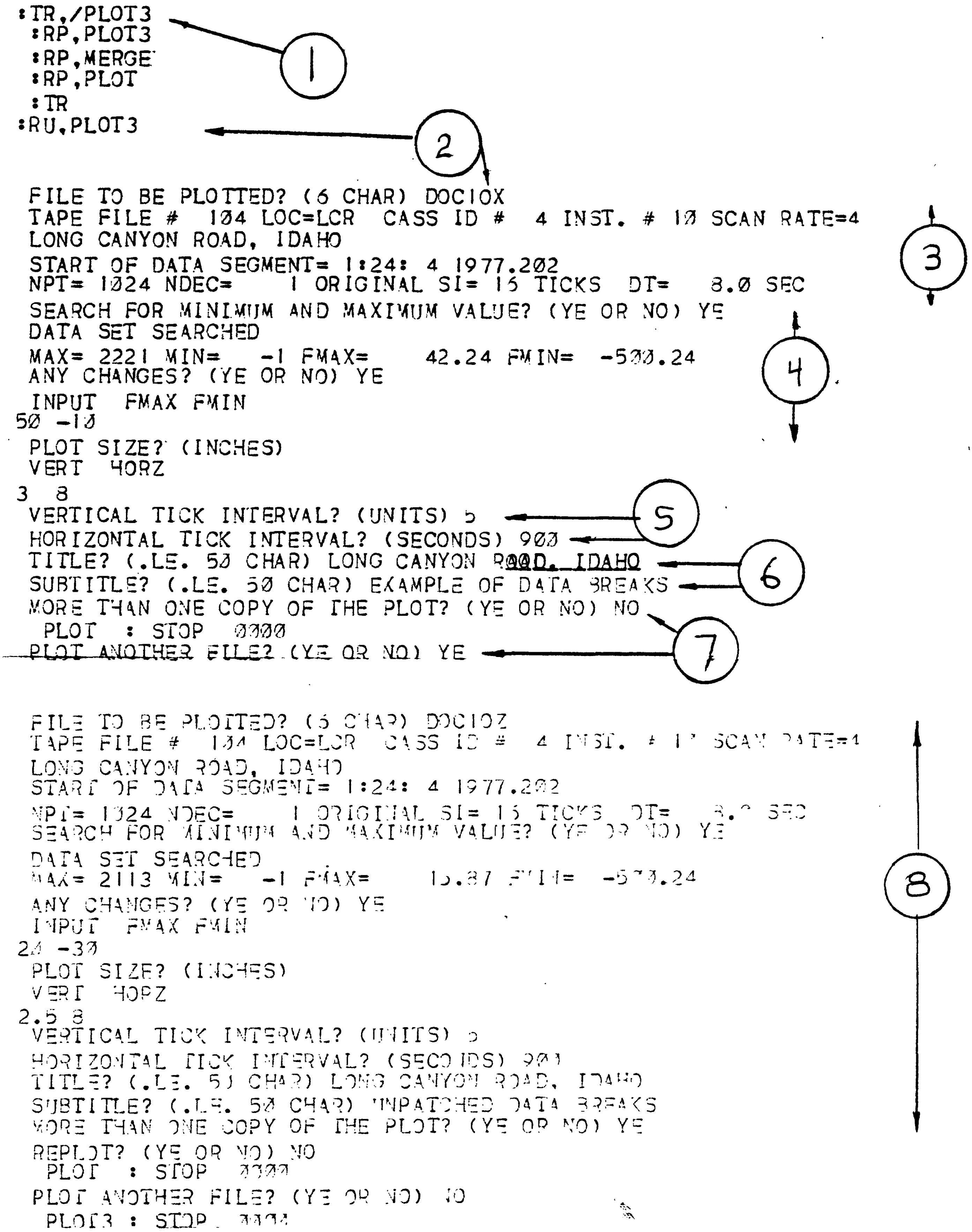
6. Data breaks are encountered after data points numbered 528 and 852 with eight and four data points missing, respectively. In both cases the user has elected to flag the data breaks and continue extracting data.
7. The actual time corresponding to the start of the data segment is printed. Notice that the segments starts four seconds after the requested time. This frequently happens since the data points do not always correspond exactly with the requested start time. The user indicated that no more data files are to be created.

PLOT3

Once a data file has been created it is often a good idea to plot it to determine if there are any peculiarities in the data. This function is performed by program PLOT3. (See Figure 16.4.)

1. This command is issued to execute the instructions stored in file /PLOT3 which restore the plotting programs. Failure to do this before running PLOT3 will cause SC∅5 scheduling errors. Issue the command :TR,\PLOT3 when no more files are to be plotted.
2. Program PLOT3 is run and file DOC10X is specified as the file to be plotted.
3. The header information is printed. Notice that the number of points in the file NPT, the decimation number NDEC (equals 1 for undecimated data), the undecimated sample interval SI in ticks (1/2 seconds), and the effective sample interval DT (DT=SI*NDEC/2) are printed.
4. The user then specifies that the data are to be searched for the minimum and maximum value. MAX and MIN are the maximum and minimum values in counts, while FMAX and FMIN are these numbers expressed in data units, which are nanoteslas for magnetic data. The user then specifies that the bounds of the plot are to be changed, and inputs new values for FMAX and FMIN.

Figure 16.4 Example of running program PLOT3



```
:TR,/PLOT3
 :RP,PLOT3
 :RP,MERGE
 :RP,PLOT
 :TR
:RU,PLOT3


 FILE TO BE PLOTTED? (6 CHAR) DOC1OX
 TAPE FILE #  1Ø4 LOC=LCR  CASS ID #  4 INST. # 1Ø SCAN RATE=4
 LONG CANYON ROAD, IDAHO
 START OF DATA SEGMENT= 1:24: 4 1977.2Ø2
 NPT= 1Ø24 NDEC=    1 ORIGINAL SI= 15 TICKS  DT=   8.Ø SEC
 SEARCH FOR MINIMUM AND MAXIMUM VALUE? (YE OR NO) YE
 DATA SET SEARCHED
 MAX= 2221 MIN=   -1 FMAX=    42.24 FMIN=  -5ØØ.24
 ANY CHANGES? (YE OR NO) YE
 INPUT  FMAX FMIN
5Ø -1Ø
 PLOT SIZE? (INCHES)
 VERT  HORZ
3  8
 VERTICAL TICK INTERVAL? (UNITS) 5
 HORIZONTAL TICK INTERVAL? (SECONDS) 9ØØ
 TITLE? (.LE. 5Ø CHAR) LONG CANYON ROAD, IDAHO
 SUBTITLE? (.LE. 5Ø CHAR) EXAMPLE OF DATA BREAKS
 MORE THAN ONE COPY OF THE PLOT? (YE OR NO) NO
  PLOT  : STOP  ØØØØ
 ~~PLOT ANOTHER FILE? (YE OR NO)~~ YE


 FILE TO BE PLOTTED? (6 CHAR) DOC1OZ
 TAPE FILE #  1Ø4 LOC=LCR  CASS ID #  4 INST. # 1Ø SCAN RATE=4
 LONG CANYON ROAD, IDAHO
 START OF DATA SEGMENT= 1:24: 4 1977.2Ø2
 NPT= 1Ø24 NDEC=    1 ORIGINAL SI= 15 TICKS  DT=   8.Ø SEC
 SEARCH FOR MINIMUM AND MAXIMUM VALUE? (YE OR NO) YE
 DATA SET SEARCHED
 MAX= 2113 MIN=   -1 FMAX=    15.87 FMIN=  -5ØØ.24
 ANY CHANGES? (YE OR NO) YE
 INPUT  FMAX FMIN
2Ø -3Ø
 PLOT SIZE? (INCHES)
 VERT  HORZ
2.5 8
 VERTICAL TICK INTERVAL? (UNITS) 5
 HORIZONTAL TICK INTERVAL? (SECONDS) 9ØØ
 TITLE? (.LE. 5Ø CHAR) LONG CANYON ROAD, IDAHO
 SUBTITLE? (.LE. 5Ø CHAR) UNPATCHED DATA BREAKS
 MORE THAN ONE COPY OF THE PLOT? (YE OR NO) YE
 REPLOT? (YE OR NO) NO
  PLOT  : STOP  ØØØØ
 PLOT ANOTHER FILE? (YE OR NO) NO
  PLOT3 : STOP  ØØØØ
```

5. A plot size of 3" by 8" is specified, along with vertical and horizontal tick-mark intervals of 5 nT and 900 seconds, respectively.
6. The title and subtitle that are to appear on the plot are input. The underlining and over-printing of the title are the result of correcting an error by using the backspace key.
7. The user specifies that only one copy of the plot is wanted. After the plot is made, the user asks to plot another file.
8. These instructions were given to create a plot of file DOC10Z.

Refer to Figure 16.5 to see the results of plotting the two files. Notice the two negative spikes on each plot. These are caused by the data break flags, which are set to a count value of -1 corresponding approximately to the negative full scale data value.

PATCH

Program PATCH is now run to interpolate across the data breaks (Figure 16.6).

1. The user specifies the name of the file to be patched, and if there are more files to process.
2. This is the printer summary for the first file patched. The numbers REC and IPT refer to the data record and location in that record, respectively, where the data break begins (FIRST) and ends (LAST). The associated DATA values are the data count values on each side of the data break. NUMB INTR tells how many data values were restored by linear interpolation, and SLOPE is the slope in counts per data point of the line used for interpolation (SLOPE=(DATA LAST - DATA FIRST)/(NUMB INTR +1)).

Figure 16.5 Example of output from program PLOT3

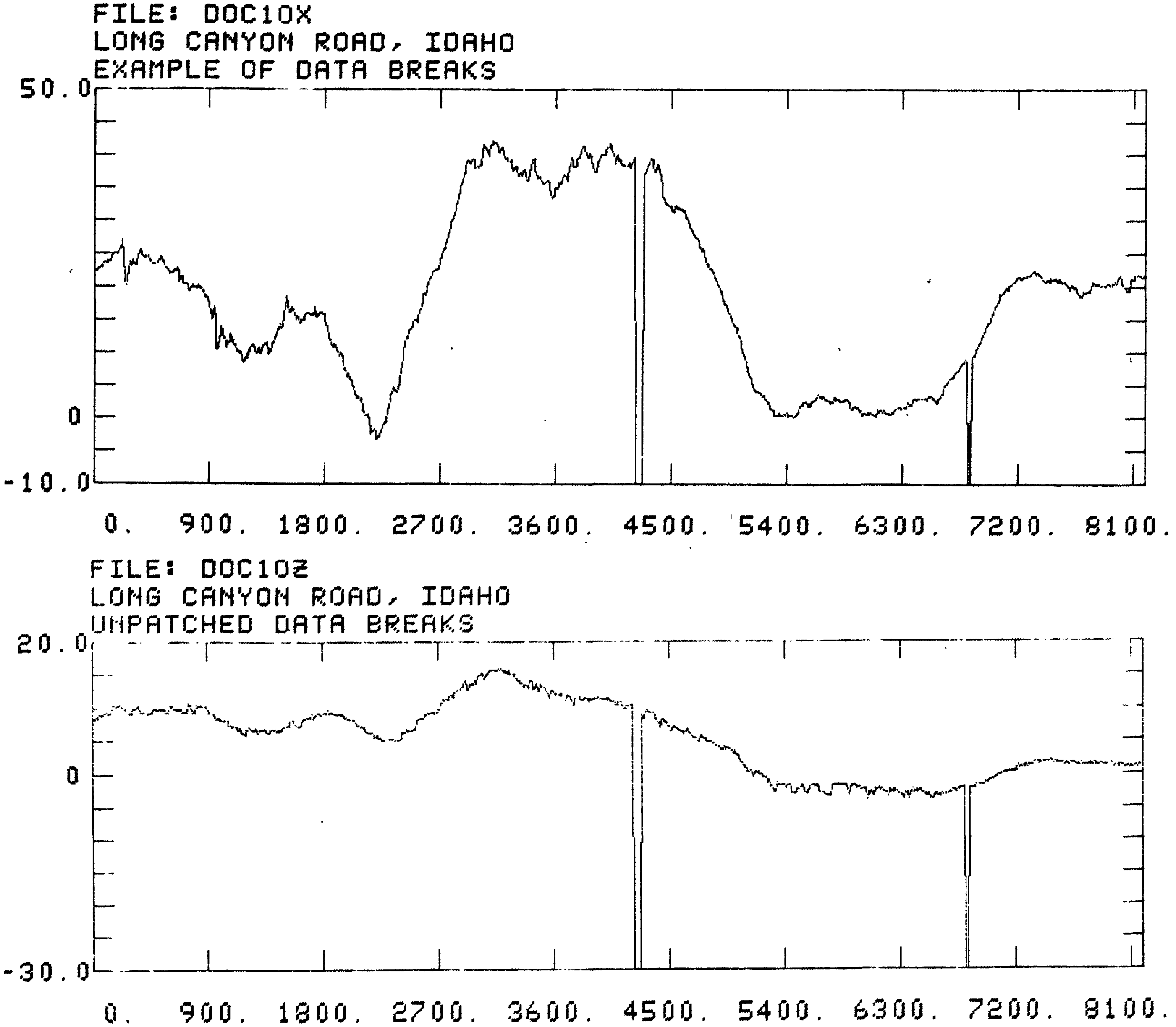
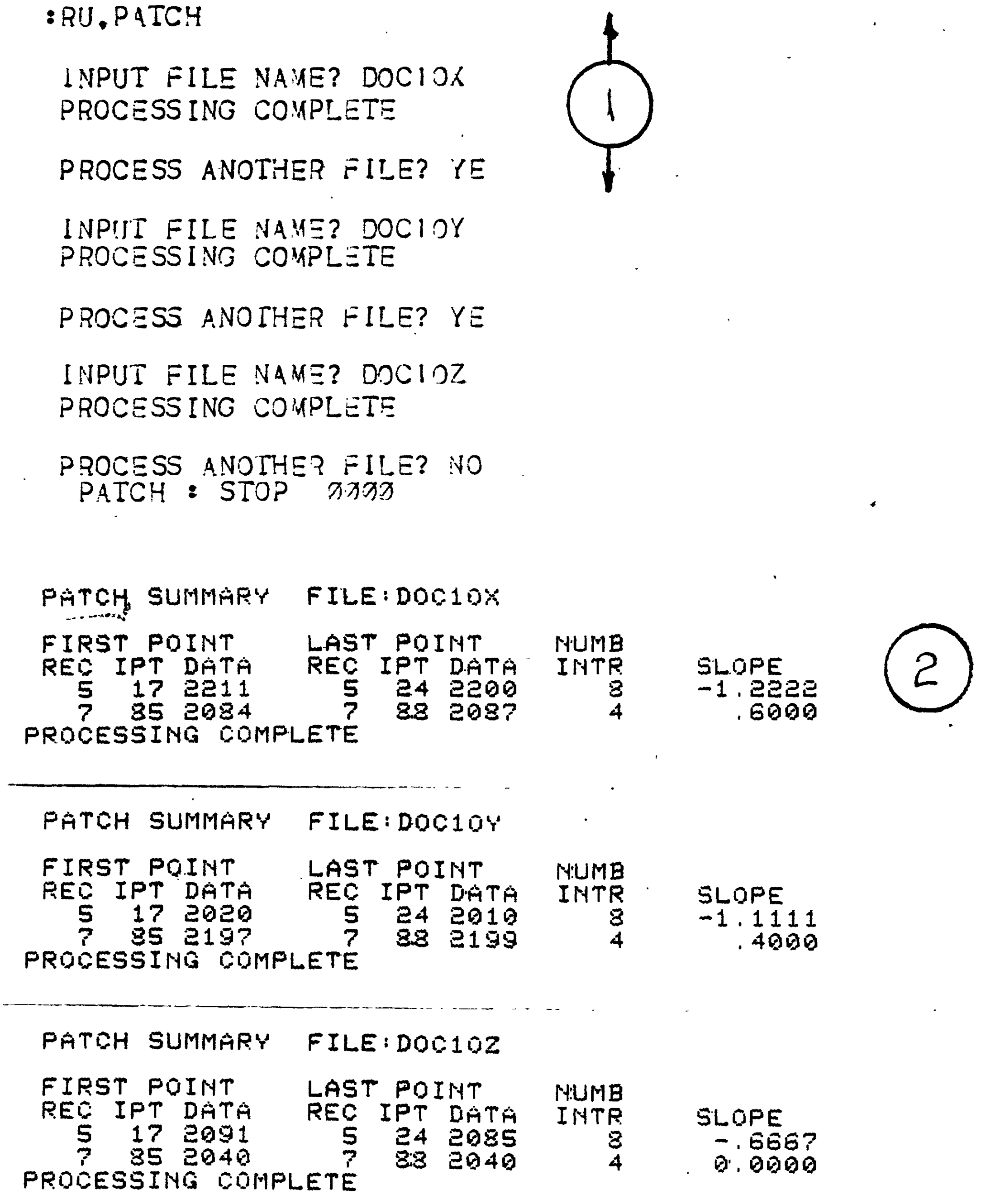


Figure 16.6 Program PATCH input and printer summary



```
:RU,PATCH

 INPUT FILE NAME? DOC10X
 PROCESSING COMPLETE

 PROCESS ANOTHER FILE? YE

 INPUT FILE NAME? DOC10Y
 PROCESSING COMPLETE

 PROCESS ANOTHER FILE? YE

 INPUT FILE NAME? DOC10Z
 PROCESSING COMPLETE

 PROCESS ANOTHER FILE? NO
  PATCH : STOP  0000
```

```
 PATCH SUMMARY  FILE:DOC10X

 FIRST POINT    LAST POINT    NUMB
 REC IPT DATA   REC IPT DATA  INTR     SLOPE
   5  17 2211     5  24 2200     8     -1.2222
   7  85 2084     7  88 2087     4        .6000
PROCESSING COMPLETE
```

```
 PATCH SUMMARY  FILE:DOC10Y

 FIRST POINT    LAST POINT    NUMB
 REC IPT DATA   REC IPT DATA  INTR     SLOPE
   5  17 2020     5  24 2010     8     -1.1111
   7  85 2197     7  88 2199     4        .4000
PROCESSING COMPLETE
```

```
 PATCH SUMMARY  FILE:DOC10Z

 FIRST POINT    LAST POINT    NUMB
 REC IPT DATA   REC IPT DATA  INTR     SLOPE
   5  17 2091     5  24 2085     8      -.6667
   7  85 2040     7  88 2040     4      0.0000
PROCESSING COMPLETE
```

Figure 16.7 shows the X and Z component data after patching.

LPBUT

The next example is the use of the low-pass filtering program LPBUT. (See Figure 16.8).

1. The user issues the run command to start the program, and then specifies the name of the file to be filtered (DOC10X).
2. The Nyquist frequency corresponding to the data sample interval is displayed.
3. The user now specifies the filter design parameters. The 3-dB point is set at 0.01 Hz, and the stop frequency is specified as 0.04 Hz. Sixty decibels of attenuation or more is desired at the stop frequency.
4. The user asks for some additional design information to be printed. A complete description of these parameters can be found in Section 4 of this report.
5. The user also requests that the frequency response of the filter be printed. The tabular output contains the angular frequency W, the true frequency FREQ, the amplitude response MAG, and the amplitude response GAIN, expressed in decibels. The quantities MAG and GAIN are given for one and two passes of the filter. The phase response is only given for a single filter pass. Since the filter is applied in a forward and backward direction, the TWO SECTION numbers are applicable, and the phase shift will be zero at all frequencies.
6. The program indicates the name of the output file, and the user decides not to filter another file.

Figure 16.7 Plot of data after using program PATCH

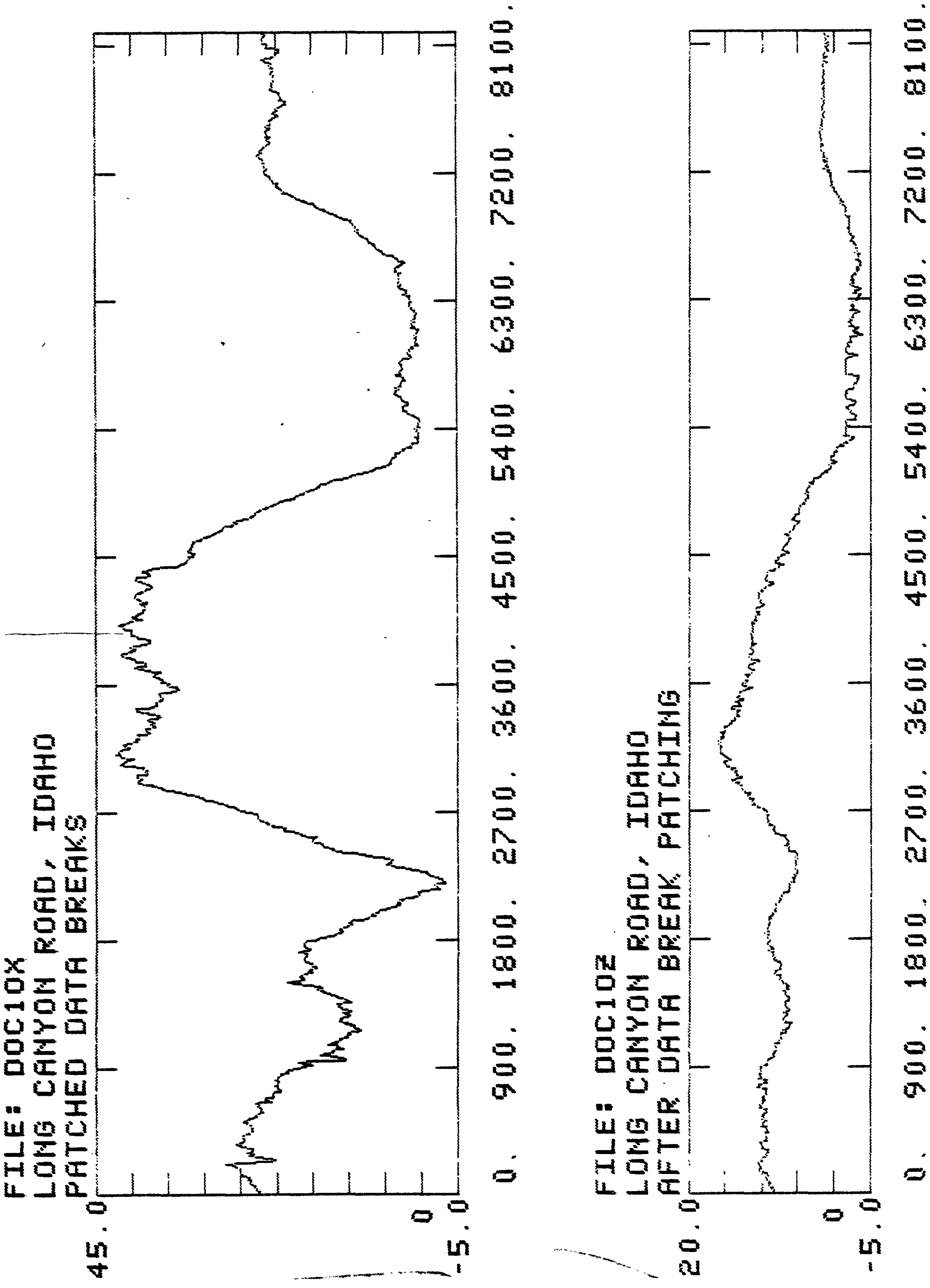
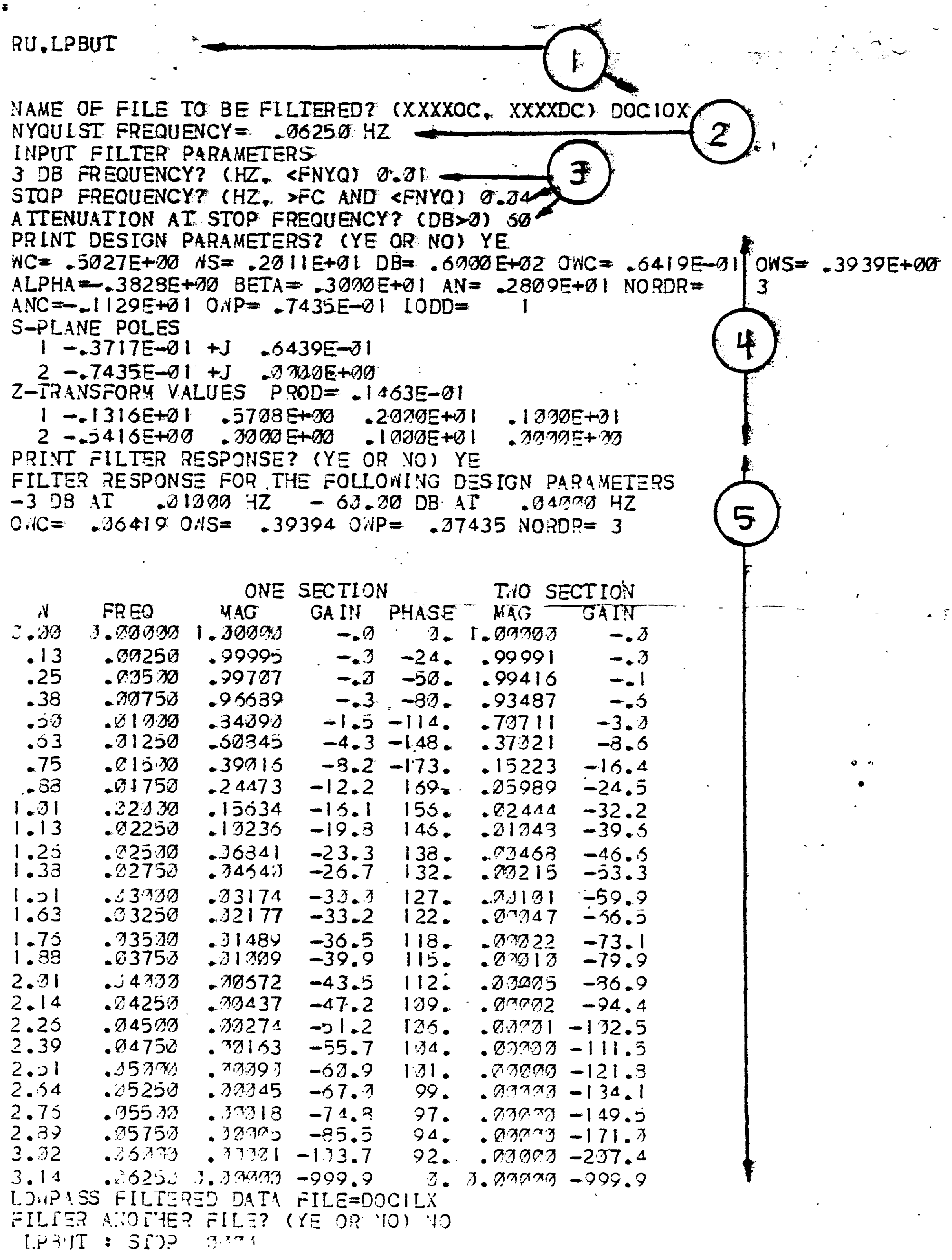


Figure 16.8 Example of running program LPBUT



```
RU,LPBUT

NAME OF FILE TO BE FILTERED? (XXXXOC, XXXXDC) DOC1OX
NYQUIST FREQUENCY=  .Ø625Ø HZ
INPUT FILTER PARAMETERS
3 DB FREQUENCY? (HZ, <FNYQ) Ø.Ø1
STOP FREQUENCY? (HZ, >FC AND <FNYQ) Ø.Ø4
ATTENUATION AT STOP FREQUENCY? (DB>Ø) 6Ø
PRINT DESIGN PARAMETERS? (YE OR NO) YE
WC= .5Ø27E+ØØ WS= .2Ø11E+Ø1 DB= .6ØØØE+Ø2 OWC= .6419E-Ø1 OWS= .3939E+ØØ
ALPHA=-.3828E+ØØ BETA= .3ØØØE+Ø1 AN= .28Ø9E+Ø1 NORDR=     3
ANC=-.1129E+Ø1 OWP= .7435E-Ø1 IODD=     1
S-PLANE POLES
  1 -.3717E-Ø1 +J  .6439E-Ø1
  2 -.7435E-Ø1 +J  .ØØØØE+ØØ
Z-TRANSFORM VALUES  PROD= .1463E-Ø1
  1 -.1316E+Ø1  .57Ø8E+ØØ  .2ØØØE+Ø1  .1ØØØE+Ø1
  2 -.5416E+ØØ  .ØØØØE+ØØ  .1ØØØE+Ø1  .ØØØØE+ØØ
PRINT FILTER RESPONSE? (YE OR NO) YE
FILTER RESPONSE FOR THE FOLLOWING DESIGN PARAMETERS
-3 DB AT   .Ø1ØØØ HZ   - 6Ø.ØØ DB AT   .Ø4ØØØ HZ
OWC=  .Ø6419 OWS=  .39394 OWP=  .Ø7435 NORDR= 3


                         ONE SECTION           TWO SECTION
  N      FREQ      MAG      GAIN  PHASE    MAG      GAIN
Ø.ØØ  Ø.ØØØØØ 1.ØØØØØ      -.Ø     Ø.  1.ØØØØØ     -.Ø
 .13    .ØØ25Ø  .99995     -.Ø   -24.   .99991     -.Ø
 .25    .ØØ5ØØ  .997Ø7     -.Ø   -5Ø.   .99416     -.1
 .38    .ØØ75Ø  .96689     -.3   -8Ø.   .93487     -.6
 .5Ø    .Ø1ØØØ  .84Ø9Ø    -1.5  -114.   .7Ø711    -3.Ø
 .63    .Ø125Ø  .6Ø845    -4.3  -148.   .37Ø21    -8.6
 .75    .Ø15ØØ  .39Ø16    -8.2  -173.   .15223   -16.4
 .88    .Ø175Ø  .24473   -12.2   169.   .Ø5989   -24.5
1.Ø1    .Ø2ØØØ  .15634   -16.1   156.   .Ø2444   -32.2
1.13    .Ø225Ø  .1Ø236   -19.8   146.   .Ø1Ø43   -39.6
1.26    .Ø25ØØ  .Ø6841   -23.3   138.   .ØØ468   -46.6
1.38    .Ø275Ø  .Ø464Ø   -26.7   132.   .ØØ215   -53.3
1.51    .Ø3ØØØ  .Ø3174   -3Ø.Ø   127.   .ØØ1Ø1   -59.9
1.63    .Ø325Ø  .Ø2177   -33.2   122.   .ØØØ47   -66.5
1.76    .Ø35ØØ  .Ø1489   -36.5   118.   .ØØØ22   -73.1
1.88    .Ø375Ø  .Ø1ØØ9   -39.9   115.   .ØØØ1Ø   -79.9
2.Ø1    .Ø4ØØØ  .ØØ672   -43.5   112.   .ØØØØ5   -86.9
2.14    .Ø425Ø  .ØØ437   -47.2   1Ø9.   .ØØØØ2   -94.4
2.26    .Ø45ØØ  .ØØ274   -51.2   1Ø6.   .ØØØØ1  -1Ø2.5
2.39    .Ø475Ø  .ØØ163   -55.7   1Ø4.   .ØØØØØ  -111.5
2.51    .Ø5ØØØ  .ØØØ9Ø   -6Ø.9   1Ø1.   .ØØØØØ  -121.8
2.64    .Ø525Ø  .ØØØ45   -67.Ø    99.   .ØØØØØ  -134.1
2.76    .Ø55ØØ  .ØØØ18   -74.8    97.   .ØØØØØ  -149.5
2.89    .Ø575Ø  .ØØØØ5   -85.5    94.   .ØØØØØ  -171.Ø
3.Ø2    .Ø6ØØØ  .ØØØØ1  -1Ø3.7    92.   .ØØØØØ  -2Ø7.4
3.14    .Ø625Ø Ø.ØØØØØ  -999.9     Ø.  Ø.ØØØØØ  -999.9
LOWPASS FILTERED DATA FILE=DOC1LX
FILTER ANOTHER FILE? (YE OR NO) NO
 LPBUT : STOP  ØØØ1
```

DECIM, MULDV, and DTRND

Figure 16.9 contains the input sequences for several programs.

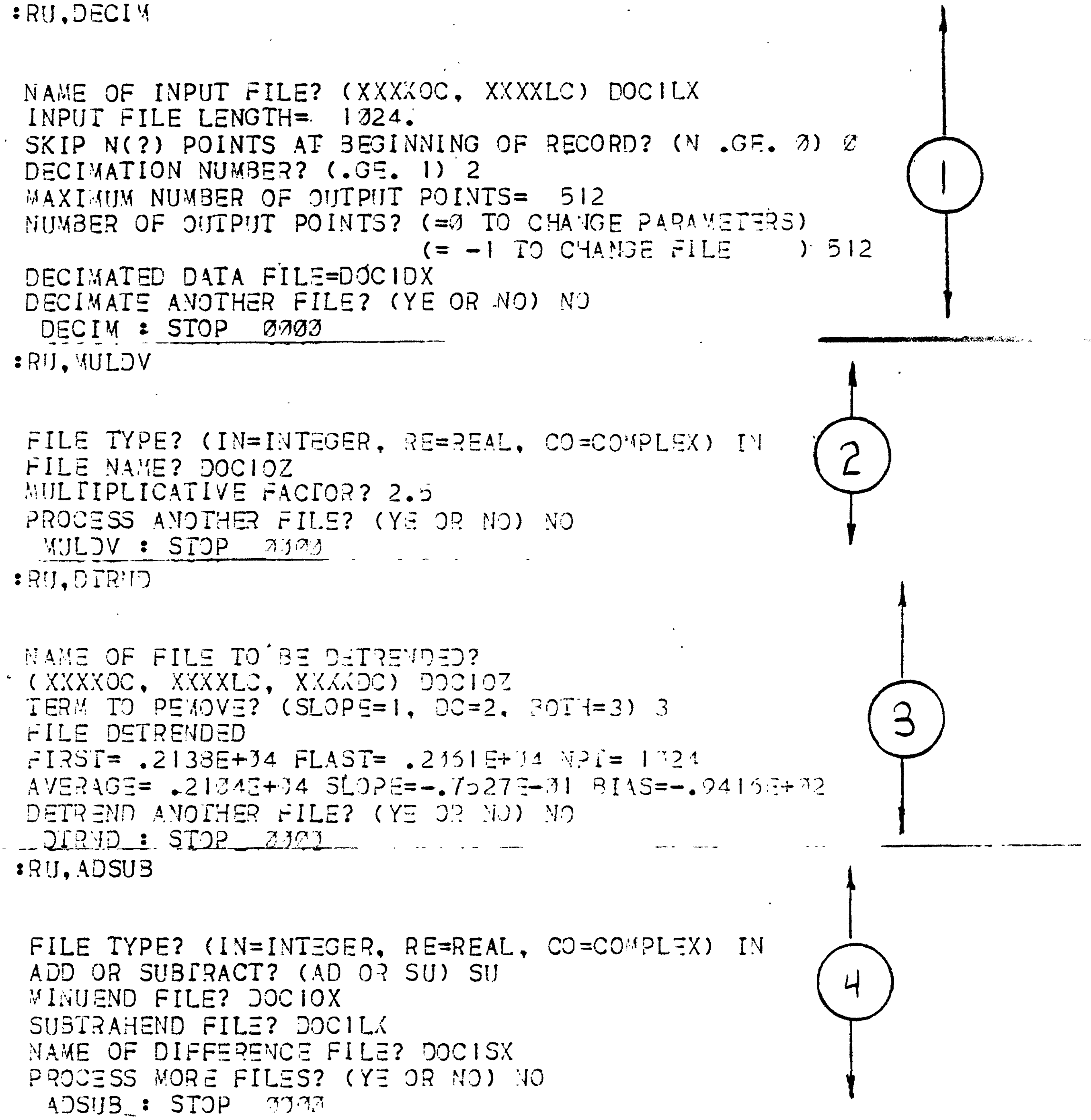
1. Program DECIM is run on file DOC1LX which has previously been low-pass filtered. The file has 1024 data points. No points are to be skipped at the beginning of the file, and every second data point is to be output. The user decides to output the maximum number of points allowed. The output file created is called DOC1DX. Figure 16.10 shows the low-pass filtered and decimated data.
2. The Z-component of magnetic field data is now multiplied by a constant using program MULDV. Figure 16.11 shows the resulting data.
3. Using program DTRND, a straight line between the end points and the DC value are removed from the Z-component data. The results are shown in Figure 16.11. The other quantities output are described in Section 8 of this report.
4. The low-pass filtered X-component data is subtracted from the original X-component data. The resulting data is put into a file called DOC1SX. These data are shown in Figure 16.12.

LSTAP and LSDSK

There are two programs that can be used to list data stored on source tape or disk files. Examples of their use is shown in Figure 16.13.

1. Program LSTAP is run to get a listing of a portion of a source tape. The program writes the header information.
2. Output is requested from record 5 subrecord 28 to record 6 subrecord 1. The user asks for the subrecord header and the data values (in counts) to be listed.

Figure 16.9 Example of running program DECIM, MULDV, DTRND, and ADSUB



```
:RU,DECIM

 NAME OF INPUT FILE? (XXXXOC, XXXXLC) DOC1LX
 INPUT FILE LENGTH=  1024.
 SKIP N(?) POINTS AT BEGINNING OF RECORD? (N .GE. 0) 0
 DECIMATION NUMBER? (.GE. 1) 2
 MAXIMUM NUMBER OF OUTPUT POINTS=  512
 NUMBER OF OUTPUT POINTS? (=0 TO CHANGE PARAMETERS)
                          (= -1 TO CHANGE FILE     ) 512
 DECIMATED DATA FILE=DOC1DX
 DECIMATE ANOTHER FILE? (YE OR NO) NO
  DECIM : STOP  0000
:RU,MULDV

 FILE TYPE? (IN=INTEGER, RE=REAL, CO=COMPLEX) IN
 FILE NAME? DOC1OZ
 MULTIPLICATIVE FACTOR? 2.5
 PROCESS ANOTHER FILE? (YE OR NO) NO
  MULDV : STOP  0000
:RU,DTRND

 NAME OF FILE TO BE DETRENDED?
 (XXXXOC, XXXXLC, XXXXDC) DOC1OZ
 TERM TO REMOVE? (SLOPE=1, DC=2, BOTH=3) 3
 FILE DETRENDED
 FIRST= .2138E+04 FLAST= .2151E+04 NPT= 1024
 AVERAGE= .2104E+04 SLOPE=-.7527E-01 BIAS=-.9416E+02
 DETREND ANOTHER FILE? (YE OR NO) NO
  DTRND : STOP  0000
:RU,ADSUB

 FILE TYPE? (IN=INTEGER, RE=REAL, CO=COMPLEX) IN
 ADD OR SUBTRACT? (AD OR SU) SU
 MINUEND FILE? DOC1OX
 SUBTRAHEND FILE? DOC1LX
 NAME OF DIFFERENCE FILE? DOC1SX
 PROCESS MORE FILES? (YE OR NO) NO
  ADSUB : STOP  0000
```

Figure 16.10 Plot of low-pass filtered and decimated data

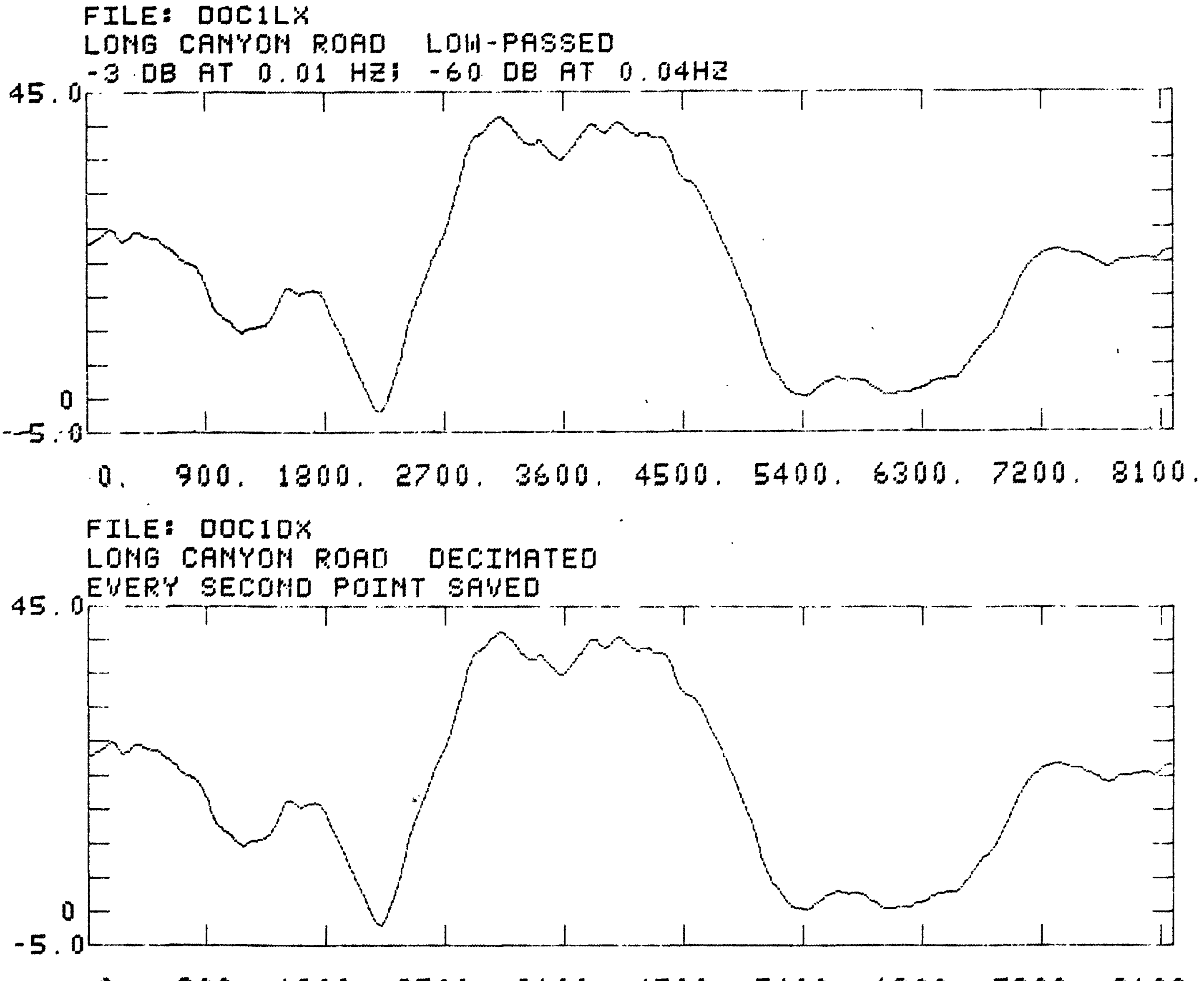


Figure 16.11 Plot of multiplied and detrended data

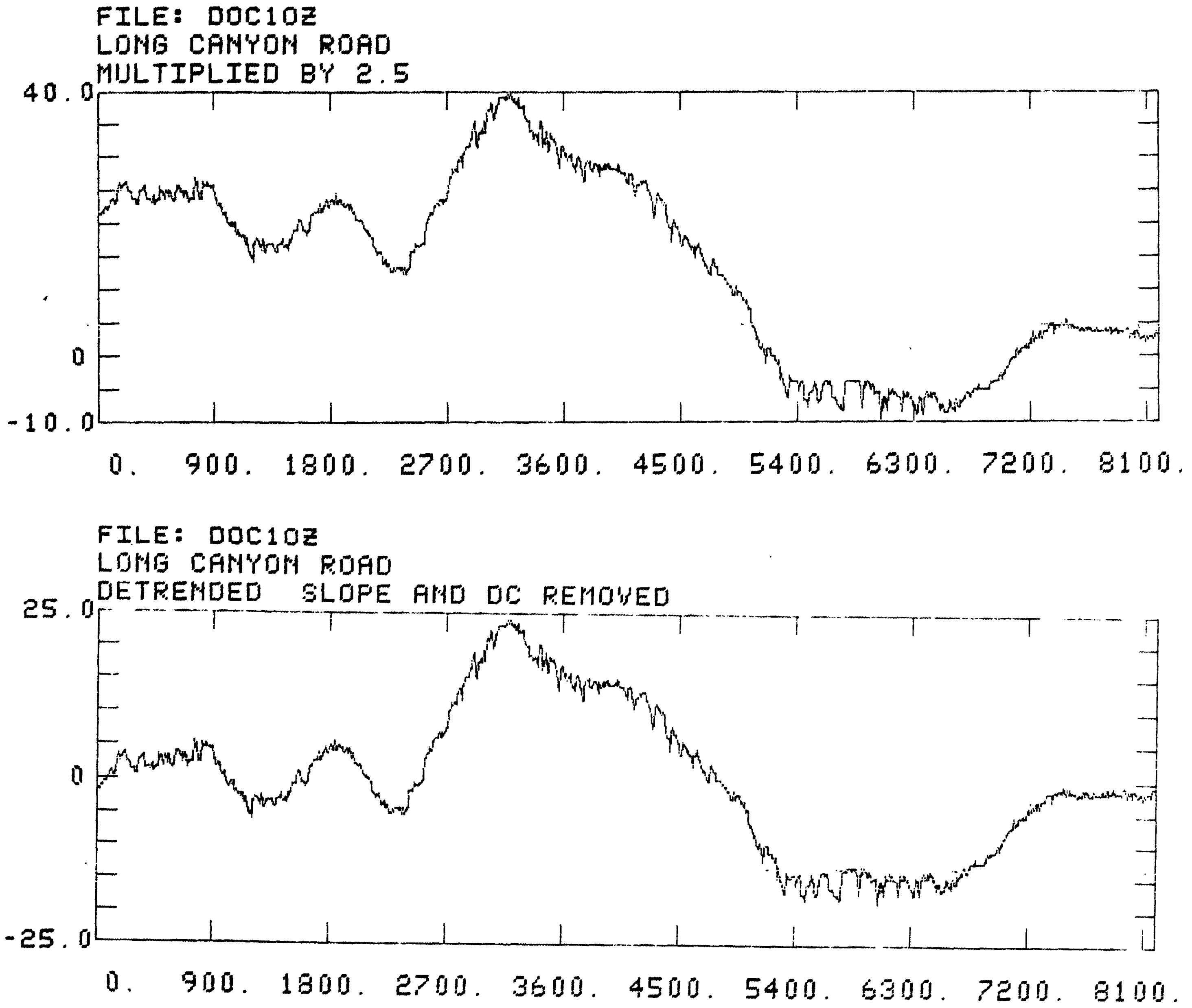


Figure 16.12 Plot of low-pass filtered data subtracted from original data.

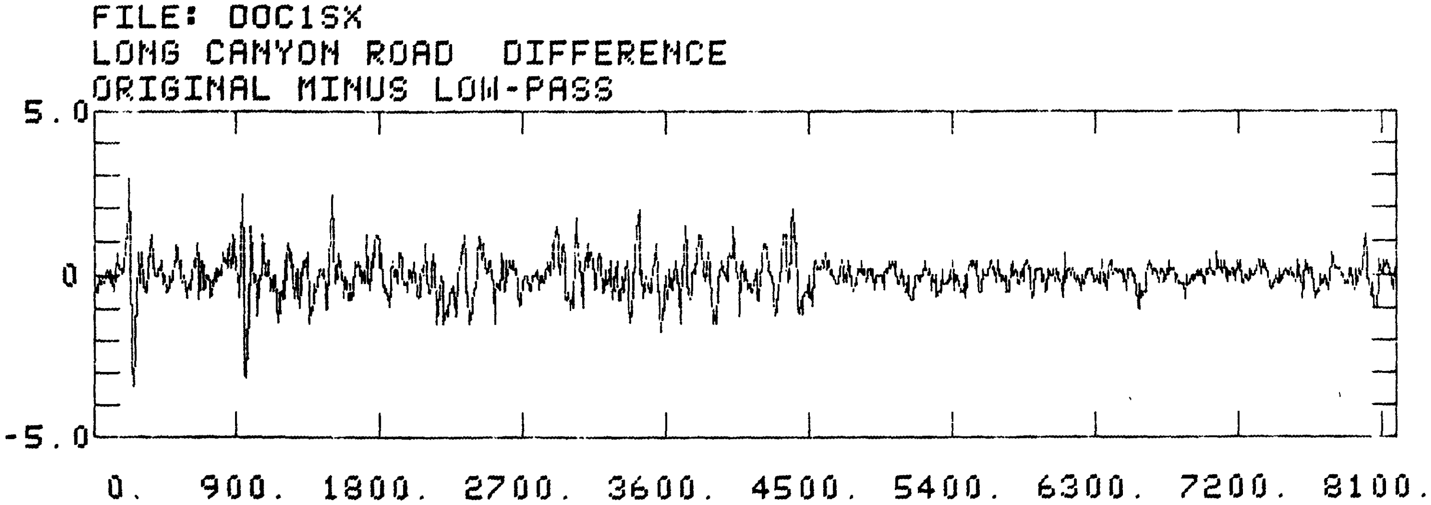
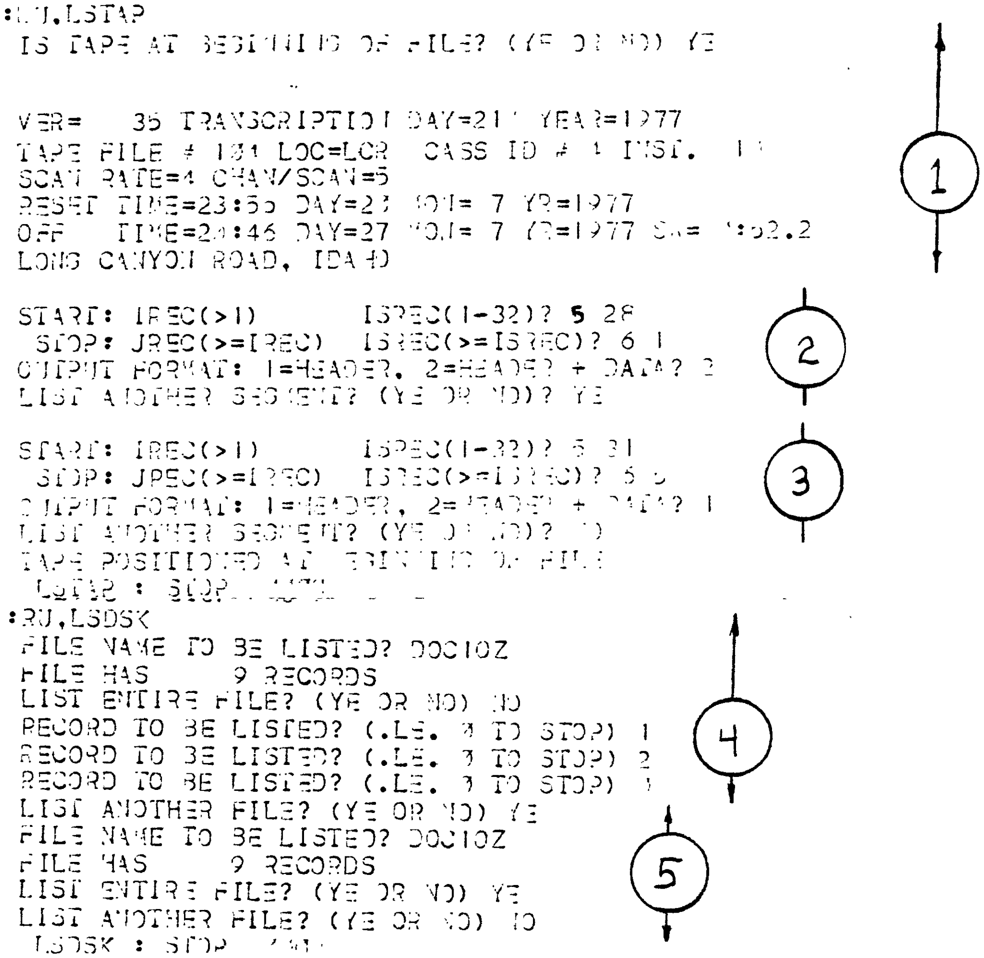


Figure 16.13 Example of running programs LSTAP and LSDSK.



```
:RU,LSTAP
 IS TAPE AT BEGINNING OF FILE? (YE OR NO) YE

 VER=    35 TRANSCRIPTION DAY=21  YEAR=1977
 TAPE FILE # 1Ø1 LOC=LCR  CASS ID # 1 INST.  1
 SCAN RATE=4 CHAN/SCAN=5
 RESET TIME=23:55 DAY=23 MON= 7 YR=1977
 OFF   TIME=20:46 DAY=27 MON= 7 YR=1977 SN= [illegible]:52.2
 LONG CANYON ROAD, IDAHO

 START: IREC(>1)        ISREC(1-32)? 5 28
  STOP: JREC(>=IREC)  ISREC(>=ISREC)? 6 1
 OUTPUT FORMAT: 1=HEADER, 2=HEADER + DATA? 2
 LIST ANOTHER SEGMENT? (YE OR NO)? YE

 START: IREC(>1)        ISREC(1-32)? 5 31
  STOP: JREC(>=IREC)  ISREC(>=ISREC)? 5 5
 OUTPUT FORMAT: 1=HEADER, 2=HEADER + DATA? 1
 LIST ANOTHER SEGMENT? (YE OR NO)? NO
 TAPE POSITIONED AT BEGINNING OF FILE
  LSTAP : STOP
:RU,LSDSK
 FILE NAME TO BE LISTED? DOC10Z
 FILE HAS      9 RECORDS
 LIST ENTIRE FILE? (YE OR NO) NO
 RECORD TO BE LISTED? (.LE. 0 TO STOP) 1
 RECORD TO BE LISTED? (.LE. 0 TO STOP) 2
 RECORD TO BE LISTED? (.LE. 0 TO STOP) 0
 LIST ANOTHER FILE? (YE OR NO) YE
 FILE NAME TO BE LISTED? DOC10Z
 FILE HAS      9 RECORDS
 LIST ENTIRE FILE? (YE OR NO) YE
 LIST ANOTHER FILE? (YE OR NO) NO
  LSDSK : STOP
```

3. A second request is now made for listing information on the same tape, however, this time only the subheader is to be printed. The output printed by LSTAP is shown in Figure 16.14, and is discussed below.
4. Program LSDSK is now run to see what was stored in file DOC10Z. This is the Z-component magnetic field taken from the source tape listed using program LSTAP. The file has nine records, and the user asks that record 1 and 2 be printed. Record 1 is the header and record 2 is the first data record.
5. All of file DOC10Z is now listed. No more files are requested to be listed. This output is shown in Figure 16.15.

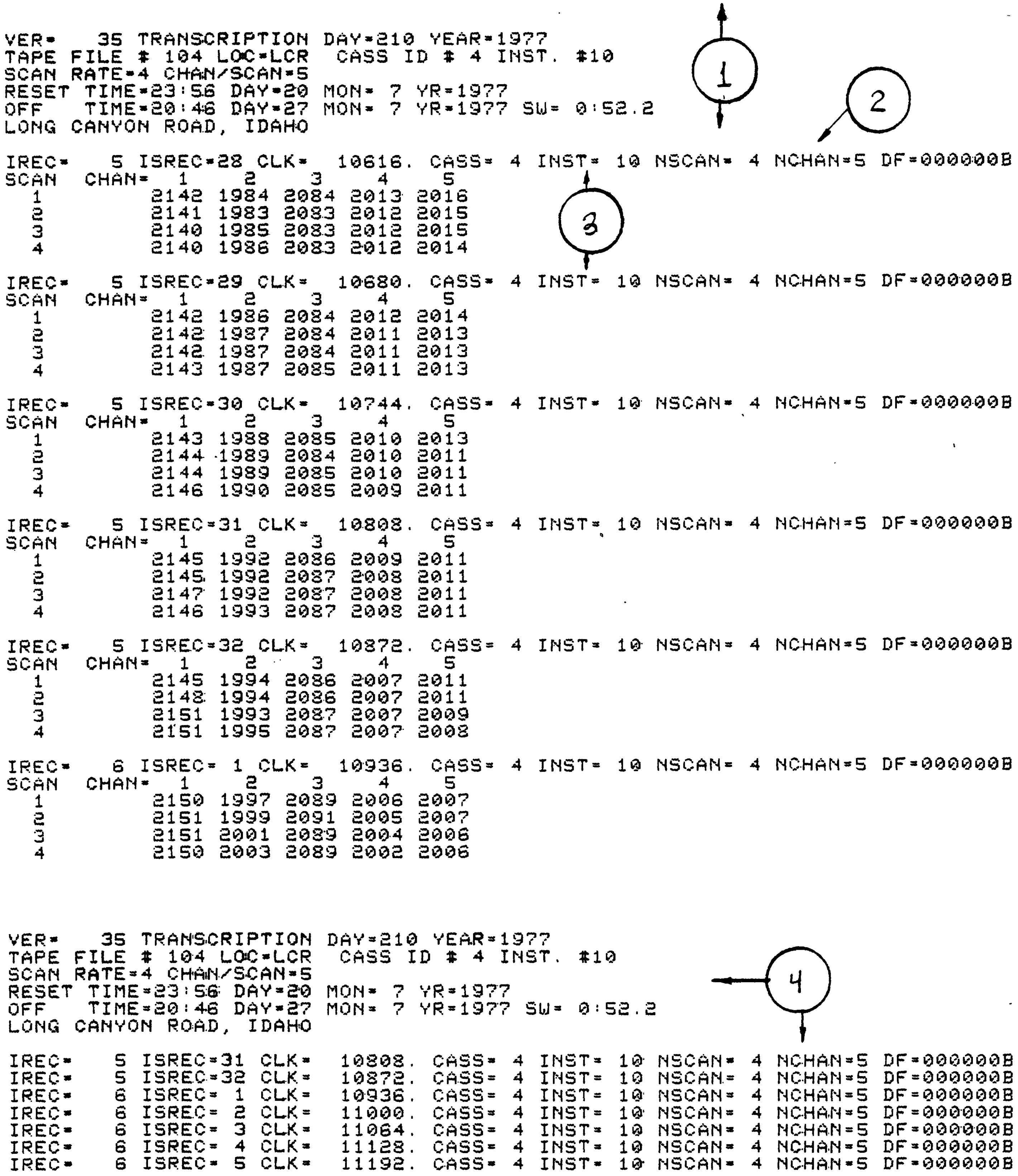
Now refer to Figure 16.14 for a description of the output printed by program LSTAP.

1. This is the standard header information.
2. This line contains the subheader for record (IREC) 5 subrecord (ISREC) 28. The clock value CLK is the time of the last scan in the subrecord.
3. This subrecord consists of four scans of five data channels. The data labeled channel 3 was put into file DOC10Z by program SLECT.
4. Program LSTAP was used again, but this time only the subheader information was requested.

Now look at Figure 16.15 to see some of the data printed by program LSDSK from file DOC10Z.

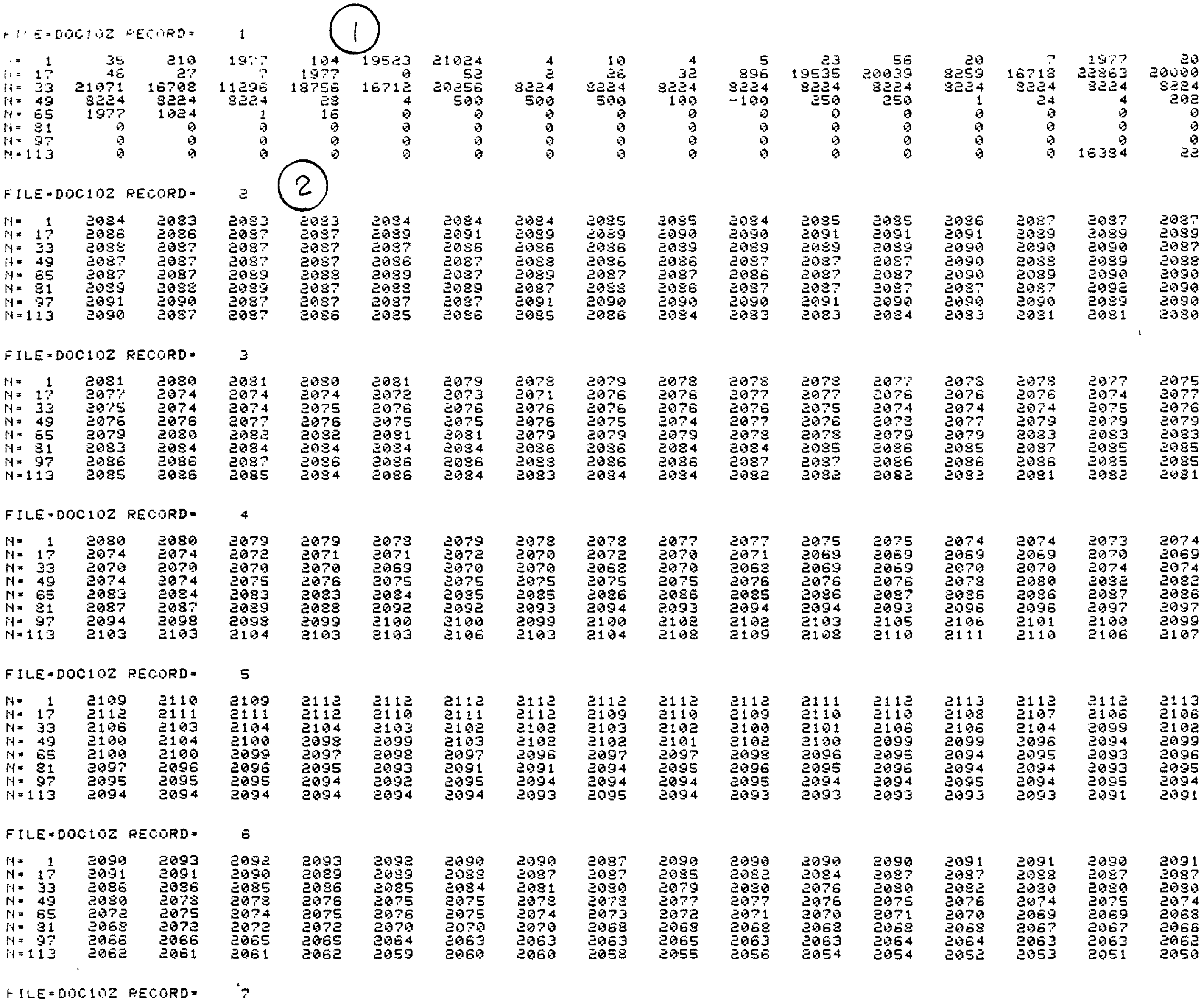
1. Record number 1 is the header record. The data are printed in integer format, 16 words per line. Some of the data are ASCII characters and floating point numbers, so they are difficult to interpret. See Appendix A -- Data Formats and File Names for a complete description.

Figure 16.14 Example of output from program LSTAP.



```
VER=   35 TRANSCRIPTION DAY=210 YEAR=1977
TAPE FILE # 104 LOC=LCR  CASS ID # 4 INST. #10
SCAN RATE=4 CHAN/SCAN=5
RESET TIME=23:56 DAY=20 MON= 7 YR=1977
OFF   TIME=20:46 DAY=27 MON= 7 YR=1977 SW= 0:52.2
LONG CANYON ROAD, IDAHO

IREC=   5 ISREC=28 CLK=  10616. CASS= 4 INST= 10 NSCAN= 4 NCHAN=5 DF=000000B
SCAN  CHAN=  1    2    3    4    5
  1         2142 1984 2084 2013 2016
  2         2141 1983 2083 2012 2015
  3         2140 1985 2083 2012 2015
  4         2140 1986 2083 2012 2014

IREC=   5 ISREC=29 CLK=  10680. CASS= 4 INST= 10 NSCAN= 4 NCHAN=5 DF=000000B
SCAN  CHAN=  1    2    3    4    5
  1         2142 1986 2084 2012 2014
  2         2142 1987 2084 2011 2013
  3         2142 1987 2084 2011 2013
  4         2143 1987 2085 2011 2013

IREC=   5 ISREC=30 CLK=  10744. CASS= 4 INST= 10 NSCAN= 4 NCHAN=5 DF=000000B
SCAN  CHAN=  1    2    3    4    5
  1         2143 1988 2085 2010 2013
  2         2144 1989 2084 2010 2011
  3         2144 1989 2085 2010 2011
  4         2146 1990 2085 2009 2011

IREC=   5 ISREC=31 CLK=  10808. CASS= 4 INST= 10 NSCAN= 4 NCHAN=5 DF=000000B
SCAN  CHAN=  1    2    3    4    5
  1         2145 1992 2086 2009 2011
  2         2145 1992 2087 2008 2011
  3         2147 1992 2087 2008 2011
  4         2146 1993 2087 2008 2011

IREC=   5 ISREC=32 CLK=  10872. CASS= 4 INST= 10 NSCAN= 4 NCHAN=5 DF=000000B
SCAN  CHAN=  1    2    3    4    5
  1         2145 1994 2086 2007 2011
  2         2148 1994 2086 2007 2011
  3         2151 1993 2087 2007 2009
  4         2151 1995 2087 2007 2008

IREC=   6 ISREC= 1 CLK=  10936. CASS= 4 INST= 10 NSCAN= 4 NCHAN=5 DF=000000B
SCAN  CHAN=  1    2    3    4    5
  1         2150 1997 2089 2006 2007
  2         2151 1999 2091 2005 2007
  3         2151 2001 2089 2004 2006
  4         2150 2003 2089 2002 2006


VER=   35 TRANSCRIPTION DAY=210 YEAR=1977
TAPE FILE # 104 LOC=LCR  CASS ID # 4 INST. #10
SCAN RATE=4 CHAN/SCAN=5
RESET TIME=23:56 DAY=20 MON= 7 YR=1977
OFF   TIME=20:46 DAY=27 MON= 7 YR=1977 SW= 0:52.2
LONG CANYON ROAD, IDAHO

IREC=   5 ISREC=31 CLK=  10808. CASS= 4 INST= 10 NSCAN= 4 NCHAN=5 DF=000000B
IREC=   5 ISREC=32 CLK=  10872. CASS= 4 INST= 10 NSCAN= 4 NCHAN=5 DF=000000B
IREC=   6 ISREC= 1 CLK=  10936. CASS= 4 INST= 10 NSCAN= 4 NCHAN=5 DF=000000B
IREC=   6 ISREC= 2 CLK=  11000. CASS= 4 INST= 10 NSCAN= 4 NCHAN=5 DF=000000B
IREC=   6 ISREC= 3 CLK=  11064. CASS= 4 INST= 10 NSCAN= 4 NCHAN=5 DF=000000B
IREC=   6 ISREC= 4 CLK=  11128. CASS= 4 INST= 10 NSCAN= 4 NCHAN=5 DF=000000B
IREC=   6 ISREC= 5 CLK=  11192. CASS= 4 INST= 10 NSCAN= 4 NCHAN=5 DF=000000B
```

Figure 16.15 Example of output from program LSDSK.



```
FILE=DOC10Z RECORD=    1           ①

N=   1      35     210    1977     104   19523   21024       4      10       4       5      23      56      20       7    1977      20
N=  17      46      27       7    1977       0      52       2      26      32     896   19535   20039    8259   16718   22863   20000
N=  33   21071   16708   11296   18756   16712   20256    8224    8224    8224    8224    8224    8224    8224    8224    8224    8224
N=  49    8224    8224    8224      28       4     500     500     500     100    -100     250     250       1      24       4     202
N=  65    1977    1024       1      16       0       0       0       0       0       0       0       0       0       0       0       0
N=  81       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
N=  97       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
N=113        0       0       0       0       0       0       0       0       0       0       0       0       0       0   16384      22

FILE=DOC10Z RECORD=    2     ②

N=   1    2084    2083    2083    2083    2084    2084    2084    2085    2085    2084    2085    2085    2086    2087    2087    2087
N=  17    2086    2086    2087    2087    2089    2091    2089    2089    2090    2090    2091    2091    2091    2089    2089    2089
N=  33    2088    2087    2087    2087    2087    2086    2086    2086    2089    2089    2089    2089    2090    2090    2090    2087
N=  49    2087    2087    2087    2087    2086    2087    2088    2086    2086    2087    2087    2087    2090    2088    2089    2088
N=  65    2087    2087    2089    2088    2089    2087    2089    2087    2087    2086    2087    2087    2090    2089    2090    2090
N=  81    2089    2088    2089    2087    2088    2089    2087    2088    2086    2087    2087    2087    2087    2087    2092    2090
N=  97    2091    2090    2087    2087    2087    2087    2091    2090    2090    2090    2091    2090    2090    2090    2089    2090
N=113     2090    2087    2087    2086    2085    2086    2085    2086    2084    2083    2083    2084    2083    2081    2081    2080

FILE=DOC10Z RECORD=    3

N=   1    2081    2080    2081    2080    2081    2079    2078    2079    2078    2078    2078    2077    2078    2078    2077    2075
N=  17    2077    2074    2074    2074    2072    2073    2071    2076    2076    2077    2077    2076    2076    2076    2074    2077
N=  33    2075    2074    2074    2075    2076    2076    2076    2076    2076    2076    2075    2074    2074    2074    2075    2076
N=  49    2076    2076    2077    2076    2075    2075    2076    2075    2074    2077    2076    2078    2077    2079    2079    2079
N=  65    2079    2080    2082    2082    2081    2081    2079    2079    2079    2078    2078    2079    2079    2083    2083    2083
N=  81    2083    2084    2084    2084    2084    2084    2086    2086    2084    2084    2085    2086    2085    2087    2085    2085
N=  97    2086    2086    2087    2086    2086    2086    2088    2086    2086    2087    2087    2086    2086    2086    2085    2085
N=113     2085    2086    2085    2084    2086    2084    2083    2084    2084    2082    2082    2082    2083    2081    2082    2081

FILE=DOC10Z RECORD=    4

N=   1    2080    2080    2079    2079    2078    2079    2078    2078    2077    2077    2075    2075    2074    2074    2073    2074
N=  17    2074    2074    2072    2071    2071    2072    2070    2072    2070    2071    2069    2069    2069    2069    2070    2069
N=  33    2070    2070    2070    2070    2069    2070    2070    2068    2070    2068    2069    2069    2070    2070    2074    2074
N=  49    2074    2074    2075    2076    2075    2075    2075    2075    2075    2076    2076    2076    2078    2080    2082    2082
N=  65    2083    2084    2083    2083    2084    2085    2085    2086    2086    2085    2086    2087    2086    2086    2087    2086
N=  81    2087    2087    2089    2088    2092    2092    2093    2094    2093    2094    2094    2093    2096    2096    2097    2097
N=  97    2094    2098    2098    2099    2100    2100    2099    2100    2102    2102    2103    2105    2106    2101    2100    2099
N=113     2103    2103    2104    2103    2103    2106    2103    2104    2108    2109    2108    2110    2111    2110    2106    2107

FILE=DOC10Z RECORD=    5

N=   1    2109    2110    2109    2112    2112    2112    2112    2112    2112    2112    2111    2112    2113    2112    2112    2113
N=  17    2112    2111    2111    2112    2110    2111    2112    2109    2110    2109    2110    2110    2108    2107    2106    2106
N=  33    2106    2103    2104    2104    2103    2102    2102    2103    2102    2100    2101    2106    2106    2104    2099    2102
N=  49    2100    2104    2100    2098    2099    2103    2102    2102    2101    2102    2100    2099    2099    2096    2094    2099
N=  65    2100    2100    2099    2097    2098    2097    2096    2097    2097    2098    2096    2095    2094    2095    2093    2096
N=  81    2097    2096    2096    2095    2093    2091    2091    2094    2095    2096    2095    2096    2094    2094    2093    2095
N=  97    2095    2095    2095    2094    2092    2095    2094    2094    2094    2095    2094    2094    2095    2094    2095    2094
N=113     2094    2094    2094    2094    2094    2094    2093    2095    2094    2093    2093    2093    2093    2093    2091    2091

FILE=DOC10Z RECORD=    6

N=   1    2090    2093    2092    2093    2092    2090    2090    2087    2090    2090    2090    2090    2091    2091    2090    2091
N=  17    2091    2091    2090    2089    2089    2088    2087    2087    2085    2082    2084    2087    2087    2088    2087    2087
N=  33    2086    2086    2085    2086    2085    2084    2081    2080    2079    2080    2076    2080    2082    2080    2080    2080
N=  49    2080    2078    2078    2076    2075    2075    2078    2073    2077    2077    2076    2075    2076    2074    2075    2074
N=  65    2072    2075    2074    2075    2076    2075    2074    2073    2072    2071    2070    2071    2070    2069    2069    2068
N=  81    2068    2072    2072    2072    2070    2070    2070    2068    2068    2068    2068    2068    2068    2067    2067    2066
N=  97    2066    2066    2065    2065    2064    2063    2063    2063    2065    2063    2063    2064    2064    2063    2063    2062
N=113     2062    2061    2061    2062    2059    2060    2060    2058    2055    2056    2054    2054    2052    2053    2051    2050

FILE=DOC10Z RECORD=    7
```

2. Record 2 is the first data record. This is data that came from channel 3 in Figure 16.14.

The examples shown above are typical of the types of operations that will be performed on geomagnetic data. The programs prompt the users for inputs, and if an unallowable input is detected, the input is requested again or an error message is written.

17. Appendix C — Program Listings

This section contains listings of all utility programs. They are presented in the order listed below. The list also shows the routine name, type of routine, and language used.

Appendix C
Program Listings

```
FTN,L
      PROGRAM SLECT,3,80
C---- PROGRAM TO EXTRACT SELECTED DATA SEGMENTS FROM
C     SOURCE TAPES AND STORE THEM AS DISC FILES.  CREATED
C     FILES ARE NAMED-
C
C                           XXXX0C
C
C     WHERE "XXXX" IS THE FOUR CHARACTER USER SUPPLIED ID,
C     AND "C" REFERS TO THE COMPONENT AS FOLLOWS:
C
C        X - HX
C        Y - HY
C        Z - HZ
C        E - EX
C        F - EY
C
C     THE "0" STANDS FOR ORIGINAL DATA.  TAPE MUST BE POSITIONED
C     AT THE BEGINNING OF THE FILE.
C
C     DATA BREAKS ARE FILLED WITH -1'S AT THE USER'S DISCRETION.
C
C     WRITTEN BY D. V. FITTERMAN, U.S.G.S., JULY 1976
C     MODIFIED 15 DECEMBER 1980
C
      DIMENSION IFILE(3),NFILE(3,5),IDCB(144,5)
      DIMENSION ISIZE(2),IHED(128),IDATA(1024),IAB(2),ICHAN(7)
      EQUIVALENCE (IAB,IA,AB),(IAB(2),IB),(IHED(127),ENPT),
     *(IHED(9),NRATE),(IHED(10),NCHAN),(IHED(52),NWORD),
     *(IHED(53),NSCAN)
      DATA LUTTY/1/,LUTAP/8/,NFILE(3,1)/2H0X/,NFILE(3,2)/2H0Y/,
     *NFILE(3,3)/2H0Z/,NFILE(3,4)/2H0E/,NFILE(3,5)/2H0F/
C
C---- IS TAPE IN PROPER PLACE?
      WRITE(LUTTY,1000)
 1000 FORMAT(//" TAPE POSITIONED AT BEGINNING OF FILE? (YE OR NO) _")
      READ(LUTTY,1010) IANS
 1010 FORMAT(3A2)
      IF(IANS .NE. 2HYE) GO TO 999
C
C---- READ HEADER AND OUTPUT
      AB=EXEC(1,100B+LUTAP,IHED,128)
C
C---- CHECK FOR EOT AND EOF
      IF(IAND(IA,240B) .NE. 0) GO TO 999
      WRITE(LUTTY,1020) (IHED(I),I=1,23),(IHED(I),I=27,51)
 1020 FORMAT(//" VER=",I5," TRANSCRIPTION DAY=",I3," YEAR=",I4/
     *" TAPE FILE #",I4," LOC=",2A2," CASS ID #",I2," INST. #",I2/
     *" SCAN RATE=",I1," CHAN/SCAN=",I1/
     *" RESET TIME=",I2,":",I2," DAY=",I2," MON=",I2," YR=",I4/
     *" OFF   TIME=",I2,":",I2," DAY=",I2," MON=",I2," YR=",I4,
     *" SW=",I2,":",I2,".",I1/1X,25A2)
C
C---- ITICK = CLOCK COUNTS PER SCAN
      ITICK=2**NRATE
      DT=NSCAN*ITICK
C
C---- COMPUTE JULIAN DAY OF FIRST AND LAST DAY
      CALL JULDY(IHED(13),IHED(14),IHED(15),IDAY)
      CALL JULDY(IHED(18),IHED(19),IHED(20),JDAY)
      WRITE(LUTTY,1030) IHED(15),IDAY,IHED(20),JDAY
 1030 FORMAT(" RESET DAY=",I4,".",I3," OFF DAY=",I4,".",I3)
C
C---- RESET END CONDITION FLAG
   20 NFLAG=0
      WRITE(LUTTY,1040)
 1040 FORMAT(/" TIME OF FIRST DATA POINT? (HOUR MIN SEC DAY YEAR)")
      READ(LUTTY,*) NHOUR,NMIN,NSEC,NDAY,NYEAR
C
C---- CHECK THAT THE START TIME IS BETWEEN RESET AND OFF TIMES
      DAYI=FPCYR(IHED(12),IHED(11),IDAY,IHED(15))
      DAYJ=FPCYR(IHED(17),IHED(16),JDAY,IHED(20))
      DAYN=FPCYR(NMIN,NHOUR,NDAY,NYEAR)
      IF((NYEAR .GT. IHED(15) .OR. (NYEAR .EQ. IHED(15)
     * .AND. NDAY .GT. IDAY) .OR. (NYEAR .EQ. IHED(15)
     * .AND. NDAY .EQ. IDAY .AND. DAYN .GE. DAYI)) .AND.
     * ((IHED(20) .GT. NYEAR) .OR. (IHED(20) .EQ. NYEAR
     * .AND. JDAY .GT. NDAY) .OR. (IHED(20) .EQ. NYEAR .AND.
     * JDAY .EQ.  NDAY .AND. DAYJ .GE. DAYN))) GO TO 30
       GO TO 20
C
C---- INPUT NUMBER OF DATA POINTS
   30 WRITE(LUTTY,1050)
 1050 FORMAT(" NUMBER OF DATA POINTS DESIRED? _")
      READ(LUTTY,*) FNPT
C
C---- COMPUTE BLOCK SIZE OF FILE (1 BLOCK= 128 WORDS)
      ISIZE(1)=FNPT/128.
      IF(128.*ISIZE(1) .LT. FNPT) ISIZE(1)=ISIZE(1)+1
C
C---- ADD ONE BLOCK FOR HEADER
      ISIZE(1)=ISIZE(1)+1
C
C---- INPUT 4 CHARACTER NAME OF FILE
   40 IFILE(1)=2H
      IFILE(2)=2H
      WRITE(LUTTY,1060)
 1060 FORMAT(" NAME OF FILE? (MUST BE 4 CHARACTERS) _")
      READ(LUTTY,1010) IFILE(1),IFILE(2)
C
C---- CHECK FOR 4 CHARACTERS
      IF(IAND(IFILE(1),177400B) .EQ. 20000B .OR.
     *IAND(IFILE(1),377B) .EQ. 40B .OR.
     *IAND(IFILE(2),177400B) .EQ. 20000B .OR.
     *IAND(IFILE(2),377B) .EQ. 40B) GO TO 40
C
C---- INPUT COMPONENTS DESIRED
      WRITE(LUTTY,1070)
 1070 FORMAT(" SELECT DESIRED CHANNELS (1=YES, 0=NO)")
C
```

```
C---- 3 CHANNELS
      IF(NCHAN .LE. 3) WRITE(LUTTY,1080)
 1080 FORMAT(" HX HY HZ")
C
C---- 5 CHANNELS
      IF(NCHAN .GE. 4) WRITE(LUTTY,1090)
 1090 FORMAT(" HX HY HZ EX EY")
      READ(LUTTY,*) (ICHAN(I),I=1,NCHAN)
C
C---- FIND BEGINNING OF DESIRED DATA SEGMENT
C
C---- DETERMINE CLOCK AT BEGINNING OF SEGMENT
      CLKS=2.*(NSEC+60.*(NMIN-IHED(12)+60.*(NHOUR-IHED(11)
     *+24.*(NDAY-IDAY+(365+LEAP(IHED(15)))*(NYEAR-IHED(15))))))
C
C---- START SEARCHING
      IREC=0
      CLKT1=-1.
      JFLAG=0
C
C---- READ A TAPE RECORD
   70 AB=EXEC(1,100B+LUTAP,IDATA,1024)
C
C---- CHECK FOR EOF
      IF(IAND(IA,200B) .NE. 0) GO TO 200
      IREC=IREC+1
C
C---- LOOP OVER SUBRECORDS
      INDEX=-NWORD
      DO 80 I=1,32
      INDEX=INDEX+NWORD
C
C---- CHECK IF BEYOND START OF SEGMENT
      CLKT2=32768.*IDATA(INDEX+1)+IDATA(INDEX+2)
C
C---- CHECK FOR CLOCK ROLL OVER
      IF(CLKT1-CLKT2 .GT. 1000000.) JFLAG=JFLAG+1
C
C---- CHECK FOR CLOCK LESS THAN START CLOCK
      CLK=CLKT2+1048576.*JFLAG
      IF(CLK .LT. CLKS) GO TO 80
      ISREC=I
      GO TO 90
   80 CLKT1=CLKT2
C
C---- READ ANOTHER TAPE SOURCE FILE RECORD
      GO TO 70
C
C---- LOCATED THE PROPER SUBRECORD, DETERMINE THE CORRECT SCAN
   90 ISCAN=NSCAN-INT(CLK-CLKS)/ITICK
      ISCAN=MAX0(1,MIN0(ISCAN,NSCAN))
      WRITE(LUTTY,1130) IREC,ISREC,ISCAN,CLK,CLKS
 1130 FORMAT(" IREC=",I5," ISREC=",I2," ISCAN=",I3,
     *" CLK=",F9.0," CLKS=",F9.0)
      IDTCK=CLK-(NSCAN-ISCAN)*ITICK-CLKS
```

```
C
C---- UPDATE STARTING TIME
      CALL TIMAD(NSEC,NMIN,NHOUR,NDAY,NYEAR,IDTCK)
C
C---- CREATE FILES
C
C---- SET UP FILE NAMES
      DO 50 I=1,NCHAN
      IF(ICHAN(I) .EQ. 0) GO TO 50
      NFILE(1,I)=IFILE(1)
      NFILE(2,I)=IFILE(2)
   55 CALL CREAT(IDCB(1,I),IER,NFILE(1,I),ISIZE,1)
      IF(IER .GE. 0) GO TO 60
C
C---- ERROR
      WRITE(LUTTY,1100) (NFILE(J,I),J=1,3),IER
 1100 FORMAT(" CREATION ERROR  FILE=",3A2," ERROR=",I3/
     *" INPUT NEW FILE NAME _")
      READ(LUTTY,1010) (NFILE(J,I),J=1,3)
      GO TO 55
   60 NBLK=IER/2
      WRITE(LUTTY,1110) (NFILE(J,I),J=1,3),NBLK
 1110 FORMAT(" CREATED FILE=",3A2," BLOCKS=",I5)
C
C---- POSITION ON SECOND RECORD OF FILE
      CALL POSNT(IDCB(1,I),IER,2,1)
   50 CONTINUE
C
C---- ENTER UNPACKING LOOP
      FIPT=0
      IOPT=1
      INDEX=INDEX+(ISCAN-1)*NCHAN+7
      CLKJ=32768.*IDATA(INDEX+1)+IDATA(INDEX+2)
      CLKI=CLKJ-DT
C
C---- LOOP OVER CHANNELS
  110 DO 120 I=1,NCHAN
      IF(ICHAN(I) .NE. 1) GO TO 120
      IDCB(IOPT+16,I)=IDATA(INDEX+I)
      IF(IOPT .LT. 128) GO TO 120
C
C---- OUTPUT BUFFER FULL, WRITE IT TO DISC
      CALL WRITF(IDCB(1,I),IER,IDCB(17,I))
  120 CONTINUE
C
C---- CHECK FOR ENOUGH DATA
      FIPT=FIPT+1.0
      IF(FIPT .EQ. FNPT) GO TO 320
      IOPT=IOPT+1
      IF(IOPT .EQ. 129) IOPT=1
C
C---- CHECK FOR ENOUGH SCANS
      IF(ISCAN .EQ. NSCAN) GO TO 130
      ISCAN=ISCAN+1
      INDEX=INDEX+NCHAN
      GO TO 110
  130 ISCAN=1
      CLKI=CLKJ
C
C---- CHECK FOR ENOUGH SUBRECORDS
      IF(ISREC .EQ. 32) GO TO 150
      ISREC=ISREC + 1
      INDEX=INDEX+NCHAN+8
C
C---- CHECK FOR DATA BREAK OR END OF DATA
  140 CLKJ=32768.*IDATA(INDEX-6)+IDATA(INDEX-5)
      IF(IDATA(INDEX-3) .GE. 0) GO TO 110
C
C---- DATA BREAK ENCOUNTERED
      IF(IDATA(INDEX-3) .LT. 0) GO TO 400
C
C---- NO MORE DATA IN SOURCE FILE
      NFLAG=2
      GO TO 310
C
C---- DATA BREAK HANDLER
C
C---- DETERMINE HOW MANY POINTS ARE MISSING
  400 DCLK=CLKJ-CLKI
      IF(DCLK .LT. -1000000.) DCLK=DCLK+1048576.
      NMISS=NSCAN*(IFIX(DCLK/DT)-1)
      WRITE(LUTTY,1300) FIPT,NMISS
 1300 FORMAT(" DATA BREAK AFTER ",F6.0," POINTS"/
     *I6," POINTS MISSING")
      WRITE(LUTTY,1310)
 1310 FORMAT(" CONTINUE SLECT WITH FLAGGED DATA HOLES? (YE OR NO) _")
      READ(LUTTY,1010) IANS
      IF(IANS .EQ. 2HYE) GO TO 410
C
C---- STOP SELECTION
      NFLAG=1
      GO TO 310
C
C---- PLACE -1 INTO OUTPUT FILES
  410 IMISS=0
  420 DO 430 I=1,NCHAN
      IF(ICHAN(I) .NE. 1) GO TO 430
      IDCB(IOPT+16,I)=-1
      IF(IOPT .LT. 128) GO TO 430
C
C---- OUTPUT BUFFER FULL, WRITE IT TO DISC
      CALL WRITE(IDCB(1,I),IER,IDCB(17,I))
  430 CONTINUE
      IMISS=IMISS+1
      FIPT=FIPT+1.0
C
C---- CHECK FOR ENOUGH DATA
      IF(FIPT .EQ. FNPT) GO TO 320
      IOPT=IOPT+1
C
```

```
C---- CHECK FOR FULL OUTPUT BUFFER
      IF(IOPT .EQ. 129) IOPT=1
C
C---- CHECK FOR MORE MISSING DATA
      IF(IMISS .LT. NMISS) GO TO 420
      GO TO 110
C
C---- READ ANOTHER RECORD
  150 AB=EXEC(1,100B+LUTAP,IDATA,1024)
C
C---- CHECK FOR EOF
      IF(IAND(IA,200B) .NE. 0) GO TO 300
      INDEX=7
      IREC=IREC+1
      ISREC=1
      ISCAN=1
      GO TO 140
C
C---- EOF ON READ
  200 WRITE(LUTTY,1140)
 1140 FORMAT(" END OF FILE ON TAPE SEARCH"/" NO FILES CREATED")
C
C---- BACKSPACE OVER FILE MARK
      CALL EXEC(3,1400B+LUTAP)
C
C---- ENOUGH DATA FOUND
  210 WRITE(LUTTY,1150)
 1150 FORMAT(" CREATE ANOTHER FILE? (YE OR NO) _")
      READ(LUTTY,1010) IANS
      IF(IANS .EQ. 2HNO) GO TO 999
C
C---- BACKSPACE A FILE
      AB=EXEC(3,1400B+LUTAP)
C
C---- FORWARD SPACE OVER EOF IF NOT AT BOT
      IF(IAND(IA,100B) .EQ. 0) CALL EXEC(3,300B+LUTAP)
C
C---- FORWARD SPACE OVER HEADER
      CALL EXEC(3,300B+LUTAP)
C
C---- ENTER PROCESS LOOP AGAIN
      GO TO 20
  300 WRITE(LUTTY,1160) FNPT,FIPT
 1160 FORMAT(" END OF FILE DURING SELECT  FNPT=",F7.0," FIPT=",F7.0)
C
C---- BACKSPACE OVER FILE MARK
      CALL EXEC(3,1400B+LUTAP)
      GO TO 320
  310 IF(NFLAG .EQ. 1) WRITE(LUTTY,1170) FNPT,FIPT
 1170 FORMAT(" DATA BREAK DURING SELECT  FNPT=",F7.0," FIPT=",F7.0)
      IF(NFLAG .EQ. 2) WRITE(LUTTY,1180) FNPT,FIPT
 1180 FORMAT(" END OF DATA DURING SELECT  FNPT=",F7.0," FIPT=",F7.0)
      FNPT=FIPT
  320 IF(IOPT .EQ. 128) GO TO 350
C
```

```
C---- ZERO END OF BUFFERS
      DO 330 I=1,NCHAN
      IF(ICHAN(I) .EQ. 0) GO TO 330
      DO 340 J=IOPT+1,128
  340 IDCB(J+16,I)=0
      CALL WRITF(IDCB(1,I),IER,IDCB(17,I))
  330 CONTINUE
C
C---- UPDATE HEADER, REWIND FILES, AND WRITE HEADER
  350 IHED(61)=NHOUR
      IHED(62)=NMIN
      IHED(63)=NSEC
      IHED(64)=NDAY
      IHED(65)=NYEAR
C
C---- SET NPT=-1 FOR VALUES GREATER THAN 32767
C     FNPT IS STORED IN IHED(127) & IHED(128)
      NPT=-1
      IF(FNPT .LE. 32767.) NPT=FNPT
      IHED(66)=NPT
      IHED(67)=1
      IHED(68)=ITICK
      WRITE(LUTTY,1190) NHOUR,NMIN,NSEC,NDAY,NYEAR
 1190 FORMAT(" NHOUR=",I2," NMIN=",I2," NSEC=",I2," NDAY=",I3,
     *" NYEAR=",I4)
C
C---- ZERO THE REST OF THE HEADER ARRAY EXCEPT FNPT IN IHED(127&128)
      DO 355 I=69,126
  355 IHED(I)=0
      DO 360 I=1,NCHAN
      IF(ICHAN(I) .EQ. 0) GO TO 360
C
C---- DETERMINE NEXT RECORD NUMBER
      CALL LOCF(IDCB(1,I),IER,NREC)
C
C---- COMPUTE NUMBER OF BLOCKS TO RETURN
      IBLK=NBLK-NREC+1
C
C---- REWIND, WRITE HEADER, AND CLOSE FILES
      CALL RWNDF(IDCB(1,I),IER)
      CALL WRITF(IDCB(1,I),IER,IHED)
      CALL CLOSE(IDCB(1,I),IER,IBLK)
  360 CONTINUE
C
C---- GO SEE IF ANOTHER FILE IS TO BE CREATED
      GO TO 210
  999 STOP
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 04248      COMMON = 00000
```

```
C
C-----&SLECS
C
C     SOURCE CODE OF SUBROUTINES FOR USE WITH SLECT
C
C     28 DECEMBER 1978
C
      SUBROUTINE JULDY(IDAY,MON,IYEAR,JDAY)
C
C---- CALCULATES JULIAN DAY FROM DAY-MONTH-YEAR
      DIMENSION NDAY(12)
      DATA NDAY/0,31,59,90,120,151,181,212,243,273,304,334/
      IL=LEAP(IYEAR)
      IF(MON .LE. 2) IL=0
      JDAY=NDAY(MON)+IDAY+IL
      RETURN
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00043      COMMON = 00000
```

```
      INTEGER FUNCTION LEAP(IYEAR)
C
C---- DETERMINES IF IYEAR IS A LEAP YEAR
C     RETURNS   LEAP=1 FOR A LEAP YEAR
C               LEAP=0 FOR ANY OTHER YEAR
      LEAP=0
      IF(MOD(IYEAR,4) .NE. 0) GO TO 20
      IF(MOD(IYEAR,100) .NE. 0) GO TO 10
      IF(MOD(IYEAR,400) .NE. 0) GO TO 20
   10 LEAP=1
   20 RETURN
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00046      COMMON = 00000
```

```
      FUNCTION FRCYR(IMIN,IHOUR,IDAY,IYEAR)
C
C---- COMPUTES THE FRACTIONAL PART OF YEAR REPRESENTED BY A
C     TIME GIVEN IN MINUTES-HOURS-JULIAN DAY
      FRCYR=((IMIN/60.+IHOUR)/24.+IDAY)/(365.+LEAP(IYEAR))
      RETURN
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00060      COMMON = 00000
```

```
      SUBROUTINE TIMAD(NSEC,NMIN,NHOUR,NDAY,NYEAR,IDIFF)
C
C---- ADDS THE DIFFERENCE IN TICKS (1/2 SECOND PULSES) TO
C     THE GIVEN TIME.  IDIFF MAY BE POSITIVE OR NEGATIVE.
      ISEC=IDIFF/2
C
C---- CHECK FOR ZERO
      IF(ISEC .EQ. 0) RETURN
      NSEC=NSEC+ISEC
      IF(NSEC .GE. 0 .AND. NSEC .LE. 59) RETURN
      IF(ISEC .LT. 0) GO TO 30
C
C---- ADDITION
   10 NSEC=NSEC-60
      NMIN=NMIN+1
      IF(NMIN .LE. 59) GO TO 20
      NMIN=0
      NHOUR=NHOUR+1
      IF(NHOUR .LE. 23) GO TO 20
      NHOUR=0
      NDAY=NDAY+1
      IF(NDAY .LE. 365+LEAP(NYEAR)) GO TO 20
      NDAY=1
      NYEAR=NYEAR+1
   20 IF(NSEC .GE. 60) GO TO 10
      RETURN
C
C---- SUBTRACTION
   30 NSEC=NSEC     60
      NMIN=NMIN-1
      IF(NMIN .GE. 0) GO TO 40
      NMIN=59
      NHOUR=NHOUR-1
      IF(NHOUR .GE. 0) GO TO 40
      NHOUR=23
      NDAY=NDAY-1
      IF(NDAY .GE. 1) GO TO 40
      NYEAR=NYEAR-1
      NDAY=365 + LEAP(NYEAR)
   40 IF(NSEC .LT. 0) GO TO 30
      RETURN
      END


 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00166      COMMON = 00000
```

```
            END$
```

```
FTN,L
      PROGRAM PATCH,3,80
C
C     PROGRAM TO PATCH HOLES IN GEOMAGNETIC VARIAITON DATA
C     FILES BY LINEAR INTERPOLAITON. DATA HOLES ARE RECOGNIZED
C     BY A DATA VALUE OF -1.  FILE TYPE MUST BE INTEGER (XXXXOC,
C     XXXXLC, OR XXXXDC).  PATCH DOES NOT CHECK FOR PROPER FILE
C     TYPE.
C
C     THIS PROGRAM USED THE FOLLOWING ASSEMBLY LANGUAGE
C     ROUTINE:
C              MOVE
C
C     WRITTEN BY D.V. FITTERMAN, U.S.G.S., AUGUST 1979
C     MODIFIED 16 DECEMBER 1980
C
      DIMENSION IDCB(144),IHED(128),IBUF(1025),INAME(3),IRECS(8),
     *IMFLG(8)
      EQUIVALENCE (FNPT,IHED(127))
      DATA NMEM/8/,IBUF/1025*0/,IRECS/8*0/,IMFLG/8*0/,
     *LUTTY/1/,LUPRT/6/
C
C---- INPUT FILE NAME
   10 WRITE(LUTTY,1000)
 1000 FORMAT(/" INPUT FILE NAME? _")
      READ(LUTTY,1010) INAME
 1010 FORMAT(3A2)
C
C---- OPEN FILE
      CALL OPEN(IDCB,IER,INAME,2)
      IF(IER .GE. 0) GO TO 20
      WRITE(LUTTY,1020) INAME,IER
 1020 FORMAT(" FILE=",3A2,"  IER=",I5)
      GO TO 180
C
C---- READ HEADER
   20 CALL READF(IDCB,IER,IHED)
C
C---- WRITE HEADING
      CALL EXEC(3,1100B+LUPRT,10)
      WRITE(LUPRT,1030) INAME
 1030 FORMAT("  PATCH SUMMARY  FILE:",3A2)
      CALL EXEC(3,1100B+LUPRT,1)
      WRITE(LUPRT,1040)
 1040 FORMAT("  FIRST POINT    LAST POINT    NUMB"/
     *"  REC IPT DATA   REC IPT DATA  INTR     SLOPE")
C
C---- DETERMINE NUMBER OF DATA POINTS
      IF(IHED(66) .NE. -1) FNPT=FLOAT(IHED(66))
C
C---- SET SOME FLAGS AND POINTERS
      INMEM=0
      FIPT=0
      LBADF=1
      IREC=2
```

```
C
C---- SET BUFFER POINTER
   30 IPT=128*INMEM+1
      CALL READF(IDCB,IER,IBUF(IPT+1),128,LEN,IREC)
C
C---- INCREMENT INMEM, RECORD NUMBER, AND MODIFIED FLAG
      INMEM=INMEM+1
      IRECS(INMEM)=IREC
      IMFLG(INMEM)=0
C
C---- INCREMENT READ POINTER
      IREC=IREC+1
C
C---- RESET RECORD COUNTER
      NPT=0
C
C---- ADVANCE BUFFER POINTER
   40 IPT=IPT+1
C
C---- ADVANCE RECORD COUNTER
      NPT=NPT+1
C
C---- ADVANCE DATA COUNTER
      FIPT=FIPT+1.
C
C---- LOOKING FOR BAD DATA?
      IF(LBADF .EQ. 0) GO TO 80
C
C---- BAD DATA FINDER
C     BAD DATA?
      IF(IBUF(IPT) .LE. -1) GO TO 50
C
C---- ALL DATA PROCESSED?
      IF(FIPT .GE. FNPT) GO TO 150
      GO TO 60
C
C---- SAVE LOCATION OF FIRST BAD DATA POINT
   50 ISTAR=IPT
C
C---- CLEAR LOOKING FOR BAD DATA FLAG
      LBADF=0
C
C---- ALL DATA PROCESSED?
      IF(FIPT .GE. FNPT) GO TO 140
C
C---- END OF RECORD
   60 IF(NPT .LE. 127) GO TO 40
C
C---- SAVE LAST DATA POINT
      IBUF(1)=IBUF(129)
C
C---- MODIFIED DATA FLAG SET?
      IF(IMFLG(1) .EQ. 0) GO TO 70
C
C---- WRITE DATA TO DISC
```

```
      CALL WRITE(IDCB,IER,IBUF(2),128,IRECS(1))
C
C---- ZERO INMEM
   70 INMEM=0
      GO TO 30
C
C---- GOOD DATA FINDER
C     GOOD DATA?
   80 IF(IBUF(IPT) .GT. -1) GO TO 90
C
C---- ALL DATA PROCESSED?
      IF(FIPT .GE. FNPT) GO TO 140
      GO TO 130
C
C---- SAVE LOCATION OF LAST BAD DATA POINT
   90 IEND=IPT-1
C
C---- INTERPOLATE DATA
      SLOPE=FLOAT(IBUF(IEND+1)-IBUF(ISTAR-1))/FLOAT(IEND-ISTAR+2)
      DO 100 I=ISTAR,IEND
  100 IBUF(I)=IBUF(ISTAR-1)+IFIX(SLOPE*FLOAT(I-ISTAR+1)+0.5)
C
C---- REPORT INTERPOLATION
      IR=IRECS(1)-1
      JR=IRECS(INMEM)-1
      IP=ISTAR-1
      JP=IEND-128*(INMEM-1)-1
      NTER=IEND-ISTAR+1
      WRITE(LUPRT,1050) IR,IP,IBUF(ISTAR-1),JR,JP,IBUF(IEND+1),
     *NTER,SLOPE
 1050 FORMAT(1X,I4,1X,I3,1X,I4,2X,I4,1X,I3,1X,I4,2X,I4,2X,F9.4)
C
C---- WRITE DATA TO DISC
      IF(INMEM .LE. 1) GO TO 120
      DO 110 I=1,INMEM-1
  110 CALL WRITE(IDCB,IER,IBUF(128*(I-1)+2),128,IRECS(I))
C
C---- SHUFFLE DATA TO HEAD OF BUFFER
      CALL MOVE(IBUF,128*(INMEM-1),IBUF,0,129)
C
C---- SHUFFLE RECORD NUMBER
      IRECS(1)=IRECS(INMEM)
      INMEM=1
C
C---- SET MODIFIED DATA FLAG
  120 IMFLG(1)=1
C
C---- SET LOOKING FOR BAD DATA FLAG
      LBADF=1
C
C---- ALL DATA PROCESSED?
      IF(FIPT .GE. FNPT) GO TO 150
C
C---- END OF RECORD
  130 IF(NPT .LE. 127) GO TO 40
```

```
C
C---- ANY ROOM IN BUFFER TO READ ANOTHER RECORD?
      IF(INMEM .LT. NMEM) GO TO 30
C
C---- DATA HOLE EXCEEDS BUFFER STORAGE
      WRITE(LUPRT,1060) NMEM
 1060 FORMAT(" DATA HOLE EXCEEDS STORAGE CAPACITY (",I2,
     *" RECORDS)"/" PROCESSING TERMINATED")
      WRITE(LUTTY,1060) NMEM
      GO TO 160
C
C---- CAN NOT EXTRAPOLATE DATA
  140 WRITE(LUPRT,1070) IRECS(1),ISTAR
 1070 FORMAT(" END OF DATA HOLE NOT FOUND: REC=",I4," IPT=",I3/
     *" PROCESSING TERMINATED")
      WRITE(LUTTY,1070) IRECS(1),ISTAR
      GO TO 160
C
C---- WRITE COMPLETION MESSAGE
  150 WRITE(LUPRT,1080)
 1080 FORMAT(" PROCESSING COMPLETE")
      WRITE(LUTTY,1080)
C
C---- MODIFIED DATA FLAG SET?
  160 IF(IMFLG(1) .EQ. 0) GO TO 170
C
C---- WRITE DATA TO DISC
      CALL WRITF(IDCB,IER,IBUF(2),128,IRECS(1))
C
C---- CLOSE FILE
  170 CALL CLOSE(IDCB)
C
C---- ADVANCE PRINTER
      CALL EXEC(3,1100B+LUPRT,10)
C
C---- PROCESS SOME MORE DATA?
  180 WRITE(LUTTY,1090)
 1090 FORMAT(/" PROCESS ANOTHER FILE? _")
      READ(LUTTY,1010) I
      IF(I .EQ. 2HYE) GO TO 10
      STOP
      END
```

```
FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


**  NO WARNINGS **  NO ERRORS **   PROGRAM = 02133      COMMON = 00000
```

```
 PAGE 0005  FTN.   9:41 AM  MON.,  9  MAR., 1981


0208         END$
```

```
 PAGE 0001
0001                    ASMB,L,T,C
MOVE   R 000005
.ENTR  X 000001
SOURC  R 000000
SORPT  R 000001
DEST   R 000002
DESPT  R 000003
NUMBR  R 000004
**  NO ERRORS PASS#1 **RTE ASMB 760924**
```

```
 PAGE 0002 #01
0001                    ASMB,L,T,C
0002  00000                       NAM MOVE,3,60
0003*
0004* MOVE   - PROGRAM TO MOVE THE CONTENTS OF ONE  ARRAY TO ANOTHER.
0005                              ENT MOVE
0006                              EXT .ENTR
0007  00000 000000  SOURC BSS 1          ADDRESS OF SOURCE BUFFER
0008  00001 000000  SORPT BSS 1          ADDRESS OF SOURCE BUFFER POINTER
0009  00002 000000  DEST  BSS 1          ADDRESS OF DESTINATION BUFFER
0010  00003 000000  DESPT BSS 1          ADDRESS OF DESTINATION BUFFER POINTER
0011  00004 000000  NUMBR BSS 1          ADDRESS OF NUMBER OF WORDS TO MOVE
0012  00005 000000  MOVE  NOP
0013  00006 016001X       JSB .ENTR      RESOLVE  INDIRECT ADDRESSES
0014  00007 000000R       DEF SOURC
0015  00010 162004R       LDA NUMBR,I
0016  00011 002003        SZA,RSS        MOVE ZERO WORDS?
0017  00012 126005R       JMP MOVE,I     YES, RETURN
0018  00013 062000R       LDA SOURC      NO, FORM SOURCE BUFFER ADDRESS
0019  00014 142001R       ADA SORPT,I
0020  00015 066002R       LDB DEST       FORM DESTINATION BUFFER ADDRESS
0021  00016 146003R       ADB DESPT,I
0022  00017 105777        MVW NUMBR,I    MOVE WORDS
      00020 100004R
      00021 000000
0023  00022 126005R       JMP MOVE,I
0024                      END MOVE
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

```
PAGE 0003                        MOVE
                       CROSS-REFERENCE SYMBOL TABLE

.ENTR   00006     00013

DESPT   00010     00021

DEST    00009     00020

MOVE    00012     00005     00017     00023     00024

NUMBR   00011     00015     00022

SORPT   00008     00019

SOURC   00007     00014     00018
```

```
FTN,L
      PROGRAM LPBUT,3,80
C
C---- PROGRAM TO LOWPASS FILTER DATA USING A RECURSIVE FILTER.
C     THE FILTER IS DESIGNED BY USING A BILINEAR TRANSFROMATION
C     ON AN ANALOG BUTTERWORTH FILTER.  THE FILTER ORDER, NORDR,
C     AND CUTOFF FREQUENCY ARE CHOSEN SUCH THAT CASCADING TWO
C     SECTIONS OF THE FILTER RESULTS IN A DESIRED CUTOFF
C     FREQUENCY AND ATTENUATION AT A SPECIFIED FREQUENCY IN THE
C     STOP BAND.  THE FILTER IS APPLIED IN THE FORWARD AND REVERSE
C     DIRECTION ON THE DATA TO INTRODUCE ZERO PHASE SHIFT.
C     THIS PROGRAM CAN FILTER ORIGINAL (XXXXOC) OR DECIMATED
C     (XXXXDC) DATA SETS.  'L' TYPE FILES CAN BE FILTERED
C     IF THEY ARE FIRST RENAMED TO 'O' OR 'D' TYPE.
C
C     WRITTEN BY D.V. FITTERMAN, U.S.G.S., SEPTEMBER 1976
C     MODIFIED 6 JANUARY 1981
C
      DIMENSION IFILE(3),ISIZE(2),IDCB(144),JDCB(144),IHED(128),
     *COEFF(4,50)
      EQUIVALENCE (IHED(127),FNPT)
      DATA LUTTY/1/,PI/3.141592/,ISIZE(2)/128/
C
C---- INPUT NAME OF FILE TO BE FILTERED
   10 WRITE(LUTTY,1000)
 1000 FORMAT(//" NAME OF FILE TO BE FILTERED? (XXXXOC, XXXXDC) _")
      READ(LUTTY,1020) IFILE
 1020 FORMAT(3A2)
C
C---- CHECK FOR 5TH CHARACTER BEING AN 'O' OR 'D'
      I=IAND(IFILE(3),177400B)
      IF(I .EQ. 47400B .OR. I .EQ. 42000B) GO TO 20
      WRITE(LUTTY,1030)
 1030 FORMAT(" FILE NAME MUST HAVE 'O' OR 'D' IN 5TH POSITION")
      GO TO 70
C
C---- CHECK FOR EXISTENCE OF FILE
   20 CALL OPEN(IDCB,IER,IFILE,2)
      IF(IER .GE. 0) GO TO 30
      WRITE(LUTTY,1040) IFILE,IER
 1040 FORMAT(" ERROR: FILE=",3A2," IER=",I5)
      GO TO 70
C
C---- READ DISC HEADER
   30 CALL READF(IDCB,IER,IHED)
C
C---- INPUT PARAMETERS
C
C     FC = CUTOFF FREQUENCY (HERTZ)
C     FS = STOP FREQUENCY (HERTZ)
C     DB = DESIRED ATTENUATION IN DB AT FS (DB>0)
C
      FNYQ=1.0/IHED(67)/IHED(68)
   35 WRITE(LUTTY,1050) FNYQ
 1050 FORMAT(" NYQUIST FREQUENCY=",F8.5," HZ"/
     *" INPUT FILTER PARAMETERS"/
     *" 3 DB FREQUENCY? (HZ, <FNYQ) _")
      READ(LUTTY,*) FC
      IF(FC .GE. FNYQ) GO TO 35
   40 WRITE(LUTTY,1060)
 1060 FORMAT(" STOP FREQUENCY? (HZ, >FC AND <FNYQ) _")
      READ(LUTTY,*) FS
C
C---- CHECK FOR STOP FREQUENCY GREATER THAN 3 DB POINT
C     AND STOP FREQUENCY GREATER THAN NYQUIST FREQUENCY
      IF(FS .LE. FC .OR. FS .GE. FNYQ) GO TO 40
      WRITE(LUTTY,1070)
 1070 FORMAT(" ATTENUATION AT STOP FREQUENCY? (DB>0) _")
      READ(LUTTY,*) DB
C
C---- SAMPLE INTERVAL
      DT=0.5/FNYQ
      WC=PI*FC/FNYQ
      WS=PI*FS/FNYQ
C
C---- DESIGN BUTTERWORTH FILTER
      CALL DFBUT(WC,WS,DB,OWC,OWS,OWP,NORDR,IODD,DT,COEF,PROD,IEXIT,
     *LUTTY)
C
C---- COMPUTE NUMBER OF TERMS IN FILTER
      NTERM=NORDR/2+IODD
      IF(IEXIT .EQ. 0) GO TO 45
      WRITE(LUTTY,1075) NTERM
 1075 FORMAT(" FILTER REQUIRES TOO MANY TERMS  NTERM=",I5," < 50")
      GO TO 35
C
C---- CHECK IF FILTER RESPONSE IS DESIRED
   45 WRITE(LUTTY,1080)
 1080 FORMAT(" PRINT FILTER RESPONSE? (YE OR NO) _")
      READ(LUTTY,1020) I
      IF(I .EQ. 2HYE) CALL RFSPN(LUTTY,COEF,PROD,NTERM,NORDR,
     *FC,FS,DB,OWC,OWS,OWP,FNYQ)
C
C---- CREATE OUTPUT FILE THE SAME LENGTH AS THE INPUT FILE
C
C---- DETERMINE LENGTH OF INPUT FILE IN SECTORS
      CALL LOCF(IDCB,IER,I,I,I,NSEC)
C
C---- GENERATE NAME OF OUTPUT FILE 'XXXXLC'
      IFILE(3)=IAND(IFILE(3),377B)
      IFILE(3)=IOR(IFILE(3),46000B)
      ISIZE(1)=NSEC/2
      CALL CREAT(JDCB,IER,IFILE,ISIZE,1)
      IF(IER .GT. 0) GO TO 50
C
C---- CREATION ERROR
      WRITE(LUTTY,1090) IFILE,IER
 1090 FORMAT(" CREATION ERROR: FILE=",3A2," IER=",I5)
      GO TO 70
C
```

```
C---- WRITE HEADER ON OUTPUT FILE
   50 CALL WRITF(JDCB,IER,IHED)
C
C---- FORWARD FILTER THE DATA
      IFLAG=1
      IREC=1
      IPT=1
      CALL FILTR(IFLAG,IREC,IPT,FNPT,IDCB,JDCB,COEF,PROD,
     *NTERM,LUTTY)
C
C---- CHECK FOR IPT=1
      IF(IPT .EQ. 1) IPT=128
C
C---- REVERSE FILTER THE DATA
      IFLAG=-1
      CALL FILTR(IFLAG,IREC,IPT,FNPT,JDCB,JDCB,COEF,PROD,
     *NTERM,LUTTY)
C
C---- REPORT FILE CREATED
      WRITE(LUTTY,1100) IFILE
 1100 FORMAT(" LOWPASS FILTERED DATA FILE=",3A2)
C
C---- CLOSE THE OUTPUT FILE
      CALL CLOSE(JDCB)
C
C---- CLOSE THE INPUT FILE
      CALL CLOSE(IDCB)
C
C---- PROCESS MORE DATA
   70 WRITE(LUTTY,1110)
 1110 FORMAT(" FILTER ANOTHER FILE? (YE OR NO) _")
      READ(LUTTY,1020) I
      IF(I .EQ. 2HYE) GO TO 10
      STOP
      END
```

```
FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


**  NO WARNINGS **  NO ERRORS **   PROGRAM = 01547      COMMON = 00000
```

```
      SUBROUTINE DEBUT(WC,WS,DB,OWC,OWS,OWP,NORDR,IODD,DT,COEF,PROD,
     *IEXIT,LUTTY)
C
C---- SUBROUTINE TO DESIGN A BUTTERWORTH FILTER AND APPLY A
C     BILINEAR TRANSFORMATION TO OBTAIN A IIR DIGITAL FILTER.
C
      DIMENSION COEF(4,50)
      DATA ALPHA/-0.3827757/,PI/3.141592/
C
C---- RESET IEXIT
      IEXIT=0
C
C---- CHECK IF DESIGN PARAMETERS ARE WANTED
      WRITE(LUTTY,1000)
 1000 FORMAT(" PRINT DESIGN PARAMETERS? (YE OR NO) _")
      READ(LUTTY,1010) IPARM
 1010 FORMAT(1A2)
C
C---- WARP FREQUENCIES
      OWC=2.*TAN(WC/2.)/DT
      OWS=2.*TAN(WS/2.)/DT
      BETA=ALOGT(10.**(DB/20.)-1.)
      AN=0.5*(BETA-ALPHA)/(ALOGT(WS)-ALOGT(WC))
      NORDR=AN
C
C---- SELECT NEXT HIGHER ORDER
      IF(NORDR .LT. AN) NORDR=NORDR+1
      IF (IPARM .EQ. 2HYE) WRITE(LUTTY,1020) WC,WS,DB,OWC,OWS,ALPHA,
     *BETA,AN,NORDR
 1020 FORMAT(" WC=",F10.4," WS=",E10.4," DB=",F10.4," OWC=",F10.4,
     *" OWS=",F10.4/
     *" ALPHA=",F10.4," BETA=",E10.4," AN=",F10.4," NORDR=",I5)
C
C---- CHECK FOR EVEN ORDER
      NORD2=NORDR/2
      IODD=NORDR-2*NORD2
C
C---- CHECK FOR TOO MANY TERMS IN FILTER
      IF(NORD2+IODD .LE. 50) GO TO 5
      IEXIT=1
      RETURN
C
C---- CALCULATE CUTOFF FREQUENCY OF SINGLE SECTION WHICH
C     PROVIDES 3 DB POINT AT OWC
    5 AN=ALOGT(OWC)-0.5*ALPHA/NORDR
      OWP=10.**AN
      IF(IPARM .EQ. 2HYE) WRITE(LUTTY,1030) AN,OWP,IODD
 1030 FORMAT(" ANC=",E10.4," OWP=",E10.4," IODD=",I5)
C
C---- COMPUTE POLES IN LEFT HALF OF S-PLANE
      AN=PI/NORDR
      DCOS=COS(AN)
      DSIN=SIN(AN)
      AN=AN*(NORDR-1)/2.
      BCOS=COS(AN)
      BSIN=SIN(AN)
C
C---- COMPUTE ONLY SECOND QUADRANT POLES IN PAIRS
      NOMES=NORD2+IODD
      DO 10 I=1,NORD2
      ACOS=BCOS*DCOS-BSIN*DSIN
      ASIN=BSIN*DCOS+BCOS*DSIN
      COEF(1,I)=ACOS*OWP
      COEF(2,I)=ASIN*OWP
      BSIN=ASIN
   10 BCOS=ACOS
C
C---- ADD POLE AT S = (-OWP,0) FOR NORDR ODD
      IF(IODD .EQ. 0) GO TO 20
      COEF(1,NORD2+1)=-OWP
      COEF(2,NORD2+1)=0.0
   20 PROD=1.0
      IF(IPARM .EQ. 2HYE) WRITE(LUTTY,1040) (J,COEF(1,J),COEF(2,J),
     *J=1,NORD2+IODD)
 1040 FORMAT(" S-PLANE POLES"/(1X,I3,1X,E10.4," +J ",E10.4))
C
C---- COMPUTE COEFFICIENTS OF Z-TRANSFORM BINOMIALS
      A2=2./DT
      A1=A2*A2
      A2=2.*A2
      DO 30 I=1,NORD2
      A=COEF(1,I)
      B=COEF(2,I)
      A3=A*A+B*B
      A4=A*A2
      A=A1-A4+A3
      COEF(1,I)=2.*(A3-A1)/A
      COEF(2,I)=(A1+A4+A3)/A
      COEF(3,I)=2.0
      COEF(4,I)=1.0
   30 PROD=PROD*A3/A
C
C---- ADD MONOMIAL FOR NORDR ODD
      IF(IODD .EQ. 0) GO TO 100
      I=NORD2+1
      A2=A2/2.
      A1=A2-COEF(1,I)
      A2=A2+COEF(1,I)
      COEF(1,I)=-A2/A1
      COEF(2,I)=0.0
      COEF(3,I)=1.0
      COEF(4,I)=0.0
      PROD=PROD*OWP/A1
  100 IF (IPARM .EQ. 2HYE) WRITE(LUTTY,1050) PROD,(J,(COEF(K,J),
     *K=1,4),J=1,NORD2+IODD)
 1050 FORMAT(" Z-TRANSFORM VALUES  PROD=",E10.4/
     *(1X,I3,1X,E10.4,1X,E10.4,1X,E10.4,1X,E10.4))
      RETURN
      END
```

```
PAGE 0006  DEBUT  9:43 AM  MON.,  9  MAR., 1981

 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00969      COMMON = 00000
```

```
      SUBROUTINE RESPN(LUTTY,COEF,PROD,NTERM,NORDR,FC,FS,DB,OWC,
     *OWS,OWP,FNYQ)
      COMPLEX A,B,H,Z,Z2
      DIMENSION COEF(4,50)
      DATA PI/3.141592/
      CMAG(Z)=SQRT(REAL(Z)**2 + AIMAG(Z)**2)
      WRITE(LUTTY,1000) FC,DB,FS,OWC,OWS,OWP,NORDR
 1000 FORMAT(" FILTER RESPONSE FOR THE FOLLOWING DESIGN PARAMETERS"/
     *" -3 DB AT ",F8.5," HZ   -",F6.2," DB AT ",F8.5," HZ"/
     *" OWC=",F8.5," OWS=",F8.5," OWP=",F8.5," NORDR=",I2//)
      WRITE(LUTTY,1010)
 1010 FORMAT(19X,"ONE SECTION",8X,"TWO SECTION"/3X,"W",4X,
     *"FREQ",5X,"MAG",4X,"GAIN  PHASE   MAG",4X,"GAIN")
      DF=FNYQ/25.
      F=-DF
      DW=PI/25.
      W=-DW
      DCOS=COS(DW)
      DSIN=SIN(DW)
      BCOS=DCOS
      BSIN=-DSIN
C
C---- LOOP OVER FREQUENCIES
      DO 10 I=1,26
      W=W+DW
      F=F+DF
      ASIN=BSIN*DCOS+BCOS*DSIN
      ACOS=BCOS*DCOS-BSIN*DSIN
      BSIN=ASIN
      BCOS=ACOS
      Z=CMPLX(ACOS,ASIN)
      H=CMPLX(PROD,0.0)
      Z2=Z*Z
C
C---- LOOP OVER NOMIALS
      DO 20 J=1,NTERM
C
C---- COMPUTE NEMERATOR AND DENOMINATOR TERMS
      A=Z2+Z*CMPLX(COEF(3,J),0.0)+CMPLX(COEF(4,J),0.0)
      B=Z2+Z*CMPLX(COEF(1,J),0.0)+CMPLX(COEF(2,J),0.0)
C
C---- CHECK FOR ZERO
      IF(CMAG(A) .LT. 1E-16) GO TO 30
C
C---- CHECK FOR POLE
      IF(CMAG(B) .LT. 1E-16) GO TO 40
   20 H=H*A/B
      GO TO 60
C
C---- ZERO HANDLER
   30 AG1=0.0
      GN1=-999.9
      GO TO 50
C
C---- POLE HANDLER
```

```
   40 AG1=9.9999
      GN1=999.9
   50 PH1=0.0
      AG2=AG1
      GN2=GN1
C
C---- NORMAL TERMINATION
   60 AG1=CMAG(H)
      IF(AG1 .GT. 1E-16) GO TO 80
      AG1=0.0
      GN1=-999.9
      PH1=0.0
      AG2=AG1
      GN2=GN1
      GO TO 70
   80 GN1=20.*ALOGT(AG1)
      PH1=57.29578*ATAN2(AIMAG(H),REAL(H))
      AG2=AG1*AG1
      GN2=2.*GN1
   70 WRITE(LUTTY,1020) W,F,AG1,GN1,PH1,AG2,GN2
 1020 FORMAT(1X,F4.2,1X,F8.5,1X,F7.5,1X,F6.1,1X,F5.0,1X,F7.5,
     *1X,F6.1)
   10 CONTINUE
      RETURN
      END


FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


**  NO WARNINGS **  NO ERRORS **   PROGRAM = 00683      COMMON = 00000
```

```
      SUBROUTINE FILTR(IFLAG,IREC,IPT,FDATA,IDCB,JDCB,COEFF,PROD,
     *NTERM,LUTTY)
      DIMENSION IDCB(144),JDCB(144),COEFF(4,50),XN(3,51)
      GDATA=0.0
C
C---- READ IREC+1 (ONE FOR HEADER)
      CALL READF(IDCB,IER,IDCB(17),128,LFN,IREC+1)
C
C---- INITIALIZE LAG VALUES
      XN(1,1)=PROD*IDCB(IPT+16)
      XN(2,1)=XN(1,1)
      XN(3,1)=XN(2,1)
      DO 10 I=1,NTERM
      F=(1.0+COEF(3,I)+COEF(4,I))/(1.0+COEF(1,I)+COEF(2,I))
      F=F*XN(1,I)
      DO 10 J=1,3
   10 XN(J,I+1)=F
C
C---- START FILTER LOOP
   30 XN(1,1)=PROD*IDCB(IPT+16)
      IF(IFBRK(DM) .LT. 0) WRITE(LUTTY,1000) IFLAG,GDATA,FDATA
 1000 FORMAT(" IFLAG=",I2," GDATA=",F7.0," FDATA=",F7.0)
C
C---- APPLY FILTER
      DO 40 I=1,NTERM
C
C---- GET INPUT
C
C---- SHUFFLE OUTPUT
      XN(3,I+1)=XN(2,I+1)
      XN(2,I+1)=XN(1,I+1)
C
C---- FILTER NEXT POINT
      XN(1,I+1)=XN(1,I)+COEFF(3,I)*XN(2,I)+COEFF(4,I)*XN(3,I)
     * -COEFF(1,I)*XN(2,I+1)-COEFF(2,I)*XN(3,I+1)
C
C---- SHUFFLE INPUT
      XN(3,I)=XN(2,I)
   40 XN(2,I)=XN(1,I)
      JDCB(IPT+16)=IFIX(XN(1,NTERM+1)+0.5)
      GDATA=GDATA+1.0
      IPT=IPT+IFLAG
C
C---- CHECK FOR FULL OUTPUT BUFFER
      IF(IFLAG .EQ. 1 .AND. IPT .LT. 129) GO TO 50
      IF(IFLAG .EQ. -1 .AND. IPT .GT. 0) GO TO 50
C
C---- WRITE RECORD TO DISC
      CALL WRITF(JDCB,IER,JDCB(17),128,IREC+1)
      IPT=128
      IF(IFLAG .EQ. 1) IPT=1
C
C---- ENOUGH DATA
   50 IF(GDATA .GE. FDATA) GO TO 60
C
```

```
C---- CHECK FOR EMPTY INPUT BUFFER
      IF(IFLAG .EQ. 1 .AND. IPT .NE. 1) GO TO 30
      IF(IFLAG .EQ. -1 .AND. IPT .NE. 128) GO TO 30
C
C---- MOVE RECORD POINTER AND READ NEXT DISC RECORD
      IREC=IREC+IFLAG
      CALL READF(IDCB,IER,IDCB(17),128,LEN,IREC+1)
      GO TO 30
C
C---- FINISH WITH THE REST OF THE DATA
C     IF NOT AT THE START OF A BUFFER WRITE IT TO DISC
   60 IF(IFLAG .EQ. 1 .AND. IPT .EQ. 1) RETURN
      IF(IFLAG .EQ. -1 .AND. IPT .EQ. 128) RETURN
      CALL WRITF(JDCB,IER,JDCB(17),128,IREC+1)
      RETURN
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00996      COMMON = 00000
```

```
0406          END$
```

```
FTN,L
      PROGRAM DECIM,3,80
C
C---- PROGRAM TO DECIMATE OR SELECT A PORTION OF AN ORIGINAL
C     DATA (XXXXOC) OR LO-PASSED DATA (XXXXLC) FILE.  THE
C     PROGRAM CAN ALSO DECIMATE 'D' TYPE FILES IF THEY ARE
C     FIRST RENAMED TO 'O' OR 'L' TYPE FILES.  THE
C     USER SPECIFIES THE INPUT FILE, THE NUMBER OF POINTS TO BE
C     SKIPPED AT THE BEGINNING OF THE FILE, THE DECIMATION
C     NUMBER (OUTPUTS EVERY N-TH POINT), AND THE NUMBER OF
C     POINTS TO BE OUTPUT.
C
C     WRITTEN BY D.V. FITTERMAN, U.S.G.S., DECEMBER 1976
C     MODIFIED 18 DECEMBER 1980
C
      DIMENSION IFILE(3),IDCB(144),JDCB(144),IHED(128),ISIZE(2)
      EQUIVALENCE (IHED(66),NPT),(IHED(127),FNPT)
      DATA LUTTY/1/,ISIZE(2)/128/
C
C---- READ NAME OF INPUT FILE
   10 WRITE(LUTTY,1000)
 1000 FORMAT(//" NAME OF INPUT FILE? (XXXXOC, XXXXLC) _")
      READ(LUTTY,1010) IFILE
 1010 FORMAT(3A2)
C
C---- TEST FOR PROPER FILE NAME
      ITEST=IAND(IFILE(3),177400B)
      IF(ITEST .EQ. 47400B .OR. ITEST .EQ. 46000B) GO TO 20
      WRITE(LUTTY,1020)
 1020 FORMAT(" FILE NAME MUST HAVE 'O' OR 'L' IN 5TH POSITION")
      GO TO 10
C
C---- CHECK FOR EXISTENCE OF FILE
   20 CALL OPEN(IDCB,IER,IFILE,2)
      IF(IER .GE. 0) GO TO 30
      WRITE(LUTTY,1030) IFILE,IER
 1030 FORMAT(" OPENING ERROR: FILE=",3A2," IER=",I5)
      GO TO 140
C
C---- READ HEADER
   30 CALL READF(IDCB,IER,IHED)
      WRITE(LUTTY,1040) FNPT
 1040 FORMAT(" INPUT FILE LENGTH=",F7.0)
C
C---- INPUT NUMBER OF POINTS TO SKIP AT BEGINNING OF RECORD
   40 WRITE(LUTTY,1050)
 1050 FORMAT(" SKIP N(?) POINTS AT BEGINNING OF RECORD? (N .GE. 0) _")
      READ(LUTTY,*) NSKIP
      IF(NSKIP .LE. -1) GO TO 40
C
C---- INPUT DECIMATION NUMBER
   42 WRITE(LUTTY,1060)
 1060 FORMAT(" DECIMATION NUMBER? (.GE. 1) _")
      READ(LUTTY,*) NDEC
      IF(NDEC .LE. 0) GO TO 42
```

```
C
C---- COMPUTE MAXIMUM ALLOWABLE NUMBER OF OUTPUT POINTS
      FNOMX=(FNPT-FLOAT(NSKIP))/FLOAT(NDEC)
      IF(FNOMX*FLOAT(NDEC) .LT. FNPT-FLOAT(NSKIP)) FNOMX=FNOMX+1.0
      NUMAX=32767
      IF(FNOMX .LT. 32767.) NUMAX=IFIX(FNOMX)
   45 WRITE(LUTTY,1070) NUMAX
 1070 FORMAT(" MAXIMUM NUMBER OF OUTPUT POINTS=",I5)
C
C---- INPUT NUMBER OF OUTPUT POINTS
      WRITE(LUTTY,1080)
 1080 FORMAT(" NUMBER OF OUTPUT POINTS? (=0 TO CHANGE PARAMETERS)"/
     *26X,"(= -1 TO CHANGE FILE     ) _")
      READ(LUTTY,*) NOUT
C
C---- CHECK IF CHANGE OF PARAMETERS IS DESIRED
      IF(NOUT .EQ. 0) GO TO 40
C
C---- CHECK IF DIFFERENT FILE IS TO BE PROCESSED
      IF(NOUT .LE. -1) GO TO 120
C
C---- CHECK FOR PARAMETER TOO LARGE
      IF(NOUT .GT. NOMAX) GO TO 45
C
C---- COMPUTE SIZE OF OUTPUT FILE
      ISIZE(1)=NOUT/128
      IF(ISIZE(1)*128 .LT. NOUT) ISIZE(1)=ISIZE(1)+1
C
C---- ADD ONE BLOCK FOR HEADER
      ISIZE(1)=ISIZE(1)+1
C
C---- FORM NAME OF OUTPUT FILE XXXXDC
      IFILE(3)=IOR(IAND(IFILE(3),377B),042000B)
C
C---- CREATE OUTPUT FILE
      CALL CREAT(JDCB,IER,IFILE,ISIZE,1)
      IF(IER .GT. 0) GO TO 50
C
C---- CREATION ERROR
      WRITE(LUTTY,1090) IFILE,IER
 1090 FORMAT(" CREATION ERROR: FILE=",3A2," IER=",I5)
      GO TO 120
C
C---- MODIFY HEADER AND WRITE TO DISC
   50 IHED(66)=NOUT
      IHED(67)=NDEC*IHED(67)
      FNPT=FLOAT(NOUT)
      CALL WRITF(JDCB,IER,IHED)
C
C---- INITIALIZE POINTERS
      JPT=1
      IPT=NSKIP+1
      IDATA=0
C
C---- READ INPUT RECORD
```

```
   60 CALL READF(IDCB,IER,IDCB(17))
   70 IF(IPT .LE. 128) GO TO 80
      IPT=IPT-128
      GO TO 60
C
C---- FILL OUTPUT BUFFER
   80 JDCB(JPT+16)=IDCB(IPT+16)
      IDATA=IDATA+1
C
C---- CHECK FOR FULL BUFFER
      IF(JPT .LT. 128) GO TO 90
C
C---- WRITE BUFFER TO DISC
      CALL WRITF(JDCB,IER,JDCB(17))
      JPT=0
   90 JPT=JPT+1
C
C---- CHECK FOR ENOUGH OUTPUT
      IF(IDATA .EQ. NOUT) GO TO 100
      IPT=IPT+NDEC
      GO TO 70
C
C---- CHECK FOR FULL BUFFER
  100 IF(JPT .EQ. 1) GO TO 110
C
C---- ZERO END OF BUFFER AND WRITE TO DISC
      DO 130 I=JPT,128
  130 JDCB(I+16)=0
      CALL WRITF(JDCB,IER,JDCB(17))
C
C---- PRINT MESSAGE ON OUTPUT FILE STATUS
  110 WRITE(LUTTY,1100) IFILE
 1100 FORMAT(" DECIMATED DATA FILE=",3A2)
C
C---- CLOSE THE FILES
      CALL CLOSE(JDCB)
  120 CALL CLOSE(IDCB)
C
C---- CHECK IF ANOTHER FILE IS TO BE PROCESSES
  140 WRITE(LUTTY,1110)
 1110 FORMAT(" DECIMATE ANOTHER FILE? (YE OR NO) _")
      READ(LUTTY,1010) ITEST
      IF(ITEST .EQ. 2HYE) GO TO 10
      STOP
      END


 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 01172       COMMON = 00000
```

```
      END$
```

```
FTN,L
      PROGRAM ADSUB,3,80
C
C---- PROGRAM TO ADD OR SUBTRACT INTEGER, REAL, OR COMPLEX FORMAT
C     DISC FILES.
C
C     WRITTEN BY D. V. FITTERMAN, U.S.G.S., JUNE 1977
C     MODIFIED 19 DECEMBER 1980
C
      DIMENSION IFILE(3),IDCB(144),IBUF(128),BUFI(64),JFILE(3),
     *JDCB(144),JBUF(128),BUFJ(64),KFILE(3),KDCB(144),KBUF(128),
     *BUFK(64),ISIZE(2),IHED(128)
      EQUIVALENCE (NPT,IHED(66)),(IBUF(1),BUFI(1)),
     *(JBUF(1),BUFJ(1)),(KBUF(1),BUFK(1))
      DATA LUTTY/1/,ISIZE(2)/128/
C
C---- TYPE OF FILES
   10 WRITE(LUTTY,1000)
 1000 FORMAT(//" FILE TYPE? (IN=INTEGER, RE=REAL, CO=COMPLEX) _")
      READ(LUTTY,1010) ITYPE
 1010 FORMAT(3A2)
      IF(ITYPE .NE. 2HIN .AND. ITYPE .NE. 2HRE .AND.
     *ITYPE .NE. 2HCO) GO TO 10
C
C---- ADDITION OR SUBTRACTION?
   20 WRITE(LUTTY,1020)
 1020 FORMAT(" ADD OR SUBTRACT? (AD OR SU) _")
      READ(LUTTY,1010) IOP
      IF(IOP .NE. 2HAD .AND. IOP .NE. 2HSU) GO TO 20
C
C---- INPUT FIRST FILE NAME
   30 IF(IOP .EQ. 2HAD) WRITE(LUTTY,1030)
 1030 FORMAT(" AUGEND FILE? _")
      IF(IOP .EQ. 2HSU) WRITE(LUTTY,1040)
 1040 FORMAT(" MINUEND FILE? _")
      READ(LUTTY,1010) IFILE
C
C---- DETERMINE COMPONENT
      CALL COMP(IFILE,LUTTY,ICOMP)
C
C---- OPEN FIRST FILE
      CALL OPEN(IDCB,IER,IFILE,2)
      IF(IER .GE. 0) GO TO 40
      WRITE(LUTTY,1050) IFILE,IER
 1050 FORMAT(" OPENING ERROR  FILE=",3A2," IER=",I4)
      GO TO 110
C
C---- INPUT SECOND FILE NAME
   40 IF(IOP .EQ. 2HAD) WRITE(LUTTY,1060)
 1060 FORMAT(" ADDEND FILE? _")
      IF(IOP .EQ. 2HSU) WRITE(LUTTY,1070)
 1070 FORMAT(" SUBTRAHEND FILE? _")
      READ(LUTTY,1010) JFILE
C
C---- DETERMINE COMPONENT
```

```
      CALL COMP(JFILE,LUTTY,ICOMP)
C
C---- OPEN SECOND INPUT FILE
      CALL OPEN(JDCB,IER,JFILE,2)
      IF(IER .GE. 0) GO TO 50
      WRITE(LUTTY,1050) JFILE,IER
      GO TO 110
C
C---- READ HEADERS
   50 CALL READF(IDCB,IER,IHED)
      CALL READF(JDCB,IER,JBUF)
C
C---- CHECK FOR HEADER DISCREPANCIES
      CALL CHECK(IHED,JBUF,IER,LUTTY)
C
C---- CHECK FOR ERRORS
      IF(IAND(IER,77B) .NE. 0) GO TO 110
C
C---- CHECK FOR WARNINGS
      IF(IAND(IER,7700B) .EQ. 0) GO TO 60
      WRITE(LUTTY,1080)
 1080 FORMAT(" CONTINUE PROCESSING? (YE OR NO) _")
      READ(LUTTY,1010) I
      IF(I .NE. 2HYE) GO TO 110
C
C---- DETERMINE LENGTH OF INPUT FILE IN SECTORS
   60 CALL LOCF(JDCB,IER,I,J,K,ISIZE(1))
C
C---- CONVERT TO BLOCKS
      ISIZE(1)=ISIZE(1)/2
C
C---- INPUT NAME OF OUTPUT FILE
      IF(IOP .EQ. 2HAD) WRITE(LUTTY,1090)
 1090 FORMAT(" NAME OF SUM FILE? _")
      IF(IOP .EQ. 2HSU) WRITE(LUTTY,1100)
 1100 FORMAT(" NAME OF DIFFERENCE FILE? _")
      READ(LUTTY,1010) KFILE
C
C---- CREATE OUTPUT FILE
      CALL CREAT(KDCB,IER,KFILE,ISIZE,1)
      IF(IER .GE. 0) GO TO 80
      WRITE(LUTTY,1110) KFILE,IER
 1110 FORMAT(" CREATION ERROR: FILE=",3A2," IER=",I4)
      GO TO 100
C
C---- COMPUTE GAIN RATIO
   80 CALL GAINS(IHED,JBUF,ICOMP,GAINR)
C
C---- WRITE HEADER TO OUTPUT FILE
      CALL WRITF(KDCB,IER,IHED)
C
C---- DO THE ARITHMETIC
      ISIGN=1
      IF(IOP .EQ. 2HSU) ISIGN=-1
C
```

```
C---- INTEGER ADDITION OR SUBTRACTION
      IF(ITYPE .EQ. 2HIN) CALL ADSBI(IDCB,IBUF,JDCB,JBUF,KDCB,
     *KBUF,GAINR,ISIGN,NPT)
C
C---- REAL ADDITION OR SUBTRACTION
      IF(ITYPE .EQ. 2HRE) CALL ADSBR(IDCB,IBUF,BUFI,JDCB,JBUF,
     *BUFJ,KDCB,KBUF,BUFK,ISIGN,NPT)
C
C---- COMPLEX ADDITION OR SUBTRACTION
      IF(ITYPE .EQ. 2HCO) CALL ADSBR(IDCB,IBUF,BUFI,JDCB,JBUF,
     *BUFJ,KDCB,KBUF,BUFK,ISIGN,2*NPT)
C
C---- CLOSE FILES
      CALL CLOSE(KDCB)
  100 CALL CLOSE(JDCB)
  110 CALL CLOSE(IDCB)
C
C---- PROCESS ANOTHER FILE
      WRITE(LUTTY,1120)
 1120 FORMAT(" PROCESS MORE FILES? (YE OR NO) _")
      READ(LUTTY,1010) I
      IF(I .EQ. 2HYE) GO TO 10
      STOP
      END


FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


**  NO WARNINGS **  NO ERRORS **   PROGRAM = 01608      COMMON = 00000
```

```
      SUBROUTINE CHECK(IHED,JHED,IER,LUTTY)
C
C---- CHECKS FOR HEADER DISCREPANCIES
      DIMENSION IHED(128),JHED(128)
      IER=0
C
C---- ERRORS
C
C---- SAME NUMBER OF POINTS?
      IF(IHED(66) .EQ. JHED(66)) GO TO 10
      IER=IER+1B
      WRITE(LUTTY,1000)
 1000 FORMAT(" ERROR: DIFFERENT NUMBER OF POINTS IN FILES")
C
C---- SAME EFFECTIVE SAMPLE INTERVAL?
   10 IF(IHED(67)*IHED(68) .EQ. JHED(67)*JHED(68)) GO TO 20
      IER=IER+2B
      WRITE(LUTTY,1010)
 1010 FORMAT(" ERROR: DIFFERENT EFFECTIVE SAMPLE INTERVAL")
C
C---- SAME PADDING VALUE?
   20 IF(IHED(69) .EQ. JHED(69)) GO TO 100
      WRITE(LUTTY,1020)
 1020 FORMAT(" ERROR: DIFFERENT PADDING VALUE")
C
C---- WARNINGS
C
C---- SAME START TIME?
  100 IF(IHED(61) .EQ. JHED(61) .AND. IHED(62) .EQ. JHED(62) .AND.
     *IHED(63) .EQ. JHED(63) .AND. IHED(64) .EQ. JHED(64) .AND.
     *IHED(65) .EQ. JHED(65)) GO TO 110
      IER=IER+100B
      WRITE(LUTTY,1100)
 1100 FORMAT(" WARNING: DIFFERENT START TIMES")
  110 RETURN
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00273      COMMON = 00000
```

```
      SUBROUTINE ADSBI(IDCB,IBUF,JDCB,JBUF,KDCB,KBUF,GAINR,
     *ISIGN,NPT)
C
C---- ROUTINE FOR ADDITION AND SUBTRACTION OF INTEGER FORMAT
C    DISC FILES
      DIMENSION IDCB(144),IBUF(128),JDCB(144),JBUF(128),KDCB(144),
     *KBUF(128)
      IDATA=0
      MX=0
   10 IPT=1
C
C---- READ RECORDS
      CALL READF(IDCB,IER,IBUF)
      CALL READF(JDCB,IER,JBUF)
C
C---- ADD DATA
   20 KBUF(IPT)=IBUF(IPT)+ISIGN*IFIX(GAINR*(JBUF(IPT)-2048))
      MX=MAX0(MX,IABS(KBUF(IPT)-2048))
      IDATA=IDATA+1
      IPT=IPT+1
      IF(IPT .LE. 128 .AND. IDATA .LT. NPT) GO TO 20
C
C---- OUTPUT SUM
      CALL WRITF(KDCB,IER,KBUF)
      IF(IDATA .LT. NPT) GO TO 10
C
C---- CHECK FOR COUNT OUTSIDE OF ALLOWABLE RANGE
      IF(MX .LE. 2048) RETURN
C
C---- READ HEADER
      CALL READF(KDCB,IER,KBUF,128,LEN,1)
      DELON=FLOAT(MX)/2048.
C
C---- ADJUST GAINS
      DO 40 I=1,5
      IF(I .LE. 3) KBUF(I+53)=IFIX(DELON*FLOAT(KBUF(I+53))+0.5)
      IF(I .GE. 4) KBUF(I+53)=IFIX(FLOAT(KBUF(I+53))/DELON+0.5)
   40 CONTINUE
C
C---- WRITE HEADER
      CALL WRITF(KDCB,IER,KBUF,128,1)
C
C---- LOOP OVER DATA
      IDATA=0
      IREC=1
   50 IREC=IREC+1
      IPT=1
      CALL READF(KDCB,IER,KBUF,128,LEN,IREC)
C
C---- ADJUST COUNTS
   60 KBUF(IPT)=IFIX(FLOAT(KBUF(IPT)-2048)/DELON+2048.5)
      IDATA=IDATA+1
      IPT=IPT+1
C
C---- CHECK FOR END OF RECORD AND DATA
```

```
      IF(IPT .LE. 128 .AND. IDATA .LT. NPT) GO TO 60
C
C---- WRITE OUTPUT
      CALL WRITF(KDCB,IER,KBUF,128,IREC)
C
C---- CHECK FOR ALL DATA PROCESSED
      IF(IDATA .LT. NPT) GO TO 50
      RETURN
      END


 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00269      COMMON = 00000
```

```
      SUBROUTINE ADSBR(IDCB,IBUF,BUFI,JDCB,JBUF,BUFJ,KDCB,KBUF,
     *BUFK,ISIGN,NPT)
C
C---- ROUTINE FOR ADDITION AND SUBTRACTION OF REAL AND COMPLEX
C     FORMAT DISC FILES
      DIMENSION IDCB(144),IBUF(128),BUFI(64),JDCB(144),JBUF(128),
     *BUFJ(64),KDCB(144),KBUF(128),BUFK(64)
      IDATA=0
   10 IPT=1
C
C---- READ RECORDS
      CALL READF(IDCB,IER,IBUF)
      CALL READF(JDCB,IER,JBUF)
C
C---- ADD DATA
   20 BUFK(IPT)=BUFI(IPT)+ISIGN*BUFJ(IPT)
      IDATA=IDATA+1
      IPT=IPT+1
      IF(IPT .LE. 64) GO TO 20
C
C---- OUTPUT SUM
      CALL WRITF(KDCB,IER,KBUF)
      IF(IDATA .LT. NPT) GO TO 10
      RETURN
      END


 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00085      COMMON = 00000
```

```
      SUBROUTINE COMP(NFILE,LUTTY,ICOMP)
C
C---- SUBROUTINE TO DETERMINE DATA COMPONENT
      DIMENSION NFILE(3)
C
C---- MASK OFF LAST CHARACTER OF NAME
      LCHAR=IAND(NFILE(3),377B)
      ICOMP=0
C
C---- HX COMPONENT (X)
      IF(LCHAR .EQ. 130B) ICOMP=1
C
C---- HY COMPONENT (Y)
      IF(LCHAR .EQ. 131B) ICOMP=2
C
C---- HZ COMPONENT (Z)
      IF(LCHAR .EQ. 132B) ICOMP=3
C
C---- EX COMPONENT (E)
      IF(LCHAR .EQ. 105B) ICOMP=4
C
C---- EY COMPONENT (F)
      IF(LCHAR .EQ. 106B) ICOMP=5
      IF(ICOMP .NE. 0) RETURN
C
C---- UNABLE TO DO AUTOMATIC COMPONENT DETERMINATION
      WRITE(LUTTY,1000)
 1000 FORMAT(" UNABLE TO DETERMINE COMPONENT")
   10 WRITE(LUTTY,1010)
 1010 FORMAT(" PLEASE SPECIFY (1=X, 2=Y, 3=Z, 4=E, 5=F) _")
      READ(LUTTY,*) ICOMP
      IF(ICOMP .LT. 1 .OR. ICOMP .GT. 5) GO TO 10
      RETURN
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00150      COMMON = 00000
```

```
      SUBROUTINE GAINS(IHED,JHED,ICOMP,GAINR)
C
C---- SUBROUTINE TO DETERMINE GAIN RATIO WHICH IS
C     DEFINED AS THE GAIN FACTOR OF THE SECOND FILE
C     DIVIDED BY THE GAIN FACTOR OF THE FIRST FILE.
      DIMENSION IHED(128),JHED(128)
      IF(ICOMP .GE. 4) GO TO 100
C
C---- MAGNETIC COMPONENT
C     GAIN FACTOR = GAMMAS PER 2048 COUNTS
      GAINR=FLOAT(JHED(ICOMP+53))/FLOAT(IHED(ICOMP+53))
      RETURN
C
C---- ELECTRIC COMPONENT
C     GAIN FACTOR = 4882.813 PER LINE LENGTH PER TELLURIC GAIN
  100 GAINR=FLOAT(IHED(ICOMP+53))*FLOAT(IHED(ICOMP+55))/
     *FLOAT(JHED(ICOMP+53))*FLOAT(JHED(ICOMP+55))
      RETURN
      END


FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

**  NO WARNINGS **  NO ERRORS **   PROGRAM = 00094      COMMON = 00000
```

```
0313          END$
```

```
PAGE 0001  FTN.   9:44 AM  MON.,  9  MAR., 1981

FTN,L
      PROGRAM MULDV,3,80
C
C---- PROGRAM TO MULTIPLY AN INTEGER, REAL, OR COMPLEX
C     FORMAT DISC FILE BY A CONSTANT.
C
C     WRITTEN BY D. V. FITTERMAN, U.S.G.S., MAY 1978
C
C    MODIFIED 18 DECEMBER 1980
C
      DIMENSION IFILE(3),IDCB(144),IBUF(128),BUFI(64),ISIZE(2),
     *IHED(128)
      EQUIVALENCE (NPT,IHED(66)),(IBUF(1),BUFI(1))
      DATA LUTTY/1/,ISIZE(2)/128/
C
C---- FILE TYPE?
   10 WRITE(LUTTY,1000)
 1000 FORMAT(//" FILE TYPE? (IN=INTEGER, RE=REAL, CO=COMPLEX) _")
      READ(LUTTY,1010) ITYPE
 1010 FORMAT(3A2)
      IF(ITYPE .NE. 2HIN .AND. ITYPE .NE. 2HRE .AND.
     *ITYPE .NE. 2HCO) GO TO 10
C
C---- FILE NAME?
      WRITE(LUTTY,1020)
 1020 FORMAT(" FILE NAME? _")
      READ(LUTTY,1010) IFILE
C
C---- OPEN FILE
      CALL OPEN(IDCB,IER,IFILE,2)
      IF(IER .GE. 0) GO TO 20
      WRITE(LUTTY,1030) IFILE,IER
 1030 FORMAT(" OPENING ERROR: FILE=",3A2," IER=",I4)
      GO TO 30
C
C---- READ HEADER
   20 CALL READF(IDCB,IER,IHED)
      WRITE(LUTTY,1040)
 1040 FORMAT(" MULTIPLICATIVE FACTOR? _")
      READ(LUTTY,*) F
      IF(ITYPE .EQ. 2HIN) GO TO 40
C
C---- SET VALUE OF N
      N=NPT
      IF(ITYPE .EQ. 2HCO) N=2*N
C
C---- PERFORM MULTIPLICATION
      CALL MULTR(IDCB,IBUF,BUFI,N,F)
C
C---- CLOSE FILE
   30 CALL CLOSE(IDCB)
      WRITE(LUTTY,1050)
 1050 FORMAT(" PROCESS ANOTHER FILE? (YE OR NO) _")
      READ(LUTTY,1010) ITYPE
      IF(ITYPE .EQ. 2HYE) GO TO 10
      STOP
C
C---- INTEGER FILE PROCESSING
   40 CALL AMX(IDCB,IBUF,NPT,MX)
C
C---- CHECK FOR NUMBER OUT OF RANGE
      DELON=F*FLOAT(MX)/2048.
      IF(ABS(DELON) .LT. 1.0) GO TO 60
C
C---- ADJUST F
      F=F/DELON
C
C---- ADJUST GAINS
      DO 50 I=1,5
      IF(I .LE. 3) IHED(I+53)=IFIX(DELON*FLOAT(IHED(I+53))+0.5)
      IF(I .GE. 4) IHED(I+53)=IFIX(FLOAT(IHED(I+53))/DELON+0.5)
   50 CONTINUE
C
C---- WRITE HEADER
   60 CALL WRITF(IDCB,IER,IHED,128,1)
C
C---- PROCESS INTEGER FILE
      CALL MULTI(IDCB,IBUF,NPT,F)
      GO TO 30
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00767      COMMON = 00000
```

```
      SUBROUTINE MULTR(IDCB,IBUF,BUFI,N,F)
C
C---- ROUTINE TO MULTIPLY REAL OR COMPLEX FILE BY A CONSTANT
      DIMENSION IDCB(144),IBUF(128),BUFI(64)
      IDATA=0
      IREC=2
   10 CALL READF(IDCB,IER,IBUF,128,LEN,IREC)
      IN=0
C
C---- MULTIPLY DATA POINT
   20 BUFI(IN+1)=F*BUFI(IN+1)
C
C---- ADVANCE COUNTERS
      IDATA=IDATA+1
      IN=IN+1
      IF(IDATA .EQ. N) GO TO 30
      IF(IN .LT. 64) GO TO 20
C
C---- WRITE RECORD TO FILE
      CALL WRITF(IDCB,IER,IBUF,128,IREC)
C
C---- ADVANCE RECORD POINTER
      IREC=IREC+1
      GO TO 10
C
C---- WRITE LAST RECORD TO FILE
   30 CALL WRITF(IDCB,IER,IBUF,128,IREC)
      RETURN
      END


 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00089      COMMON = 00000
```

```
      SUBROUTINE AMX(IDCB,IBUF,NPT,MX)
C
C---- FIND MAXIMUM DEVIATION OF POINTS
      DIMENSION IDCB(144),IBUF(128)
C
C---- SET COUNTERS
      IDATA=0
      MX=0
   10 IPT=1
      CALL READF(IDCB,IER,IBUF)
   20 MX=MAX0(MX,IABS(IBUF(IPT)-2048))
      IDATA=IDATA+1
      IPT=IPT+1
C
C---- CHECK FOR END OF RECORD
      IF(IPT .LE. 128) GO TO 20
C
C---- CHECK FOR ALL DATA SEARCHED
      IF(IDATA .LT. NPT) GO TO 10
      RETURN
      END


 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00061      COMMON = 00000
```

```
      SUBROUTINE MULTI(IDCB,IBUF,NPT,F)
C
C---- ROUTINE TO MULTIPLY INTEGER FILE BY A CONSTANT
      DIMENSION IDCB(144),IBUF(128)
      IDATA=0
      IREC=1
C
C---- LOOP OVER DATA
   10 IPT=1
      IREC=IREC+1
      CALL READF(IDCB,IER,IBUF,128,LEN,IREC)
C
C---- MULTIPLY DATA BY CONSTANT
   20 IBUF(IPT)=IFIX(F*FLOAT(IBUF(IPT)-2048)+2048.5)
      IDATA=IDATA+1
      IPT=IPT+1
C
C---- CHECK FOR END OF RECORD AND DATA
      IF(IPT .LE. 128 .AND. IDATA .LT. NPT) GO TO 20
C
C---- WRITE DATA TO FILE
      CALL WRITF(IDCB,IER,IBUF,128,IREC)
C
C---- CHECK FOR ALL DATA PROCESSED
      IF(IDATA .LT. NPT) GO TO 10
      RETURN
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00087      COMMON = 00000
```

```
0158          END$
```

```
FTN,L
      PROGRAM DTRND,3,80
C
C---- ROUTINE TO REMOVE STRAIGHT LINE BETWEEN THE ENDPOINTS
C     AND/OR THE AVERAGE (DC) LEVEL. REMOVAL OF THE LINEAR
C     SLOPE TERM INSURES PERIODICITY.  INPUT FILES MAY BE
C     ORIGINAL (XXXXOC), DECIMATED (XXXXDC), OR FILTERED(XXXXLC).
C     OUTPUT IS RETURNED TO THE INPUT FILE.
C
C     WRITTEN BY D.V. FITTERMAN, U.S.G.S., JANUARY 1977
C     MODIFIED 24 DECEMBER 1980
C
      DIMENSION IDCB(144),IFILE(3),IBUF(128),IHED(128)
      EQUIVALENCE (IHED(66),NPT),(IHED(127),FNPT)
      DATA LUTTY/1/
C
C---- INPUT NAME OF FILE TO BE DETRENDED
   10 WRITE(LUTTY,1000)
 1000 FORMAT(//" NAME OF FILE TO BE DETRENDED?"/
     *" (XXXXOC, XXXXLC, XXXXDC) _")
      READ(LUTTY,1010) IFILE
 1010 FORMAT(3A2)
C
C---- TEST FOR 'O', 'L', OR 'D' IN 5TH POSITION
      I=IAND(IFILE(3),177400B)
      IF(I .EQ. 47400B .OR. I .EQ. 46000B .OR. I .EQ. 42000B) GO TO 20
      WRITE(LUTTY,1020)
 1020 FORMAT(" FILE NAME MUST HAVE 'O', 'L', OR 'D' IN 5TH POSITION")
      GO TO 10
C
C---- SEE IF FILE EXISTS
   20 CALL OPEN(IDCB,IER,IFILE,2)
      IF(IER .GE. 0) GO TO 30
      WRITE(LUTTY,1030) IFILE,IER
 1030 FORMAT(" OPENING ERROR: FILE=",3A2," IER=",I5)
      GO TO 110
C
C---- READ HEADER
   30 CALL READF(IDCB,IER,IHED)
C
C---- CHECK FOR FILE LENGTH > 32767
      IF(NPT .GT. 0) GO TO 35
      WRITE(LUTTY,1035) FNPT
 1035 FORMAT(" FILE LENGTH=",F7.0," > 32767")
C
C---- CLOSE THE FILE
      CALL CLOSE(IDCB)
      GO TO 110
C
C---- READ FIRST RECORD AND GET FIRST DATA WORD
   35 CALL READF(IDCB,IER,IBUF)
      FIRST=IBUF(1)
      SUM=0.0
      IPT=1
      IDATA=1
   40 SUM=SUM+IBUF(IPT)
      IDATA=IDATA+1
      IPT=IPT+1
C
C---- TEST FOR ENOUGH DATA
      IF(IDATA .GT. NPT) GO TO 50
C
C---- TEST FOR EMPTY BUFFER
      IF(IPT .LE. 128) GO TO 40
C
C---- READ ANOTHER RECORD AND RESET POINTER
      CALL READF(IDCB,IER,IBUF)
      IPT=1
      GO TO 40
   50 FLAST=IBUF(IPT-1)
C
C---- COMPUTE AVERAGE AND SLOPE
      SUM=SUM/FLOAT(NPT)
      SLOPE=(FLAST-FIRST)/FLOAT(NPT-1)
   60 WRITE(LUTTY,1040)
 1040 FORMAT(" TERM TO REMOVE? (SLOPE=1, DC=2, BOTH=3) _")
      READ(LUTTY,*) ITERM
      IF(ITERM .GT. 3 .OR. ITERM .LT. 1) GO TO 60
C
C---- SELECT SLOPE TERM
      IF(ITERM .EQ. 2) SLOPE =0.0
C
C---- SELECT BIAS TERM
      BIAS=0.0
      IF(ITERM .EQ. 2) BIAS=2048.-SUM
      IF(ITERM .EQ. 3) BIAS=2048.-SUM+0.5*FLOAT(NPT-1)*SLOPE
C
C---- BEGIN DETREND LOOP
      IPT=1
      IDATA=0
      IREC=1
   70 CALL READF(IDCB,IER,IBUF,128,I,IREC+1)
   80 IBUF(IPT)=IBUF(IPT)-IFIX(SLOPE*IDATA-BIAS+0.5)
      IDATA=IDATA+1
      IPT=IPT+1
      IF(IPT .LE. 128) GO TO 90
C
C---- EMPTY THE BUFFER
      CALL WRITF(IDCB,IER,IBUF,128,IREC+1)
      IREC=IREC+1
      IPT=1
C
C---- CHECK FOR ENOUGH DATA PROCESSED
      IF(IDATA .LT. NPT) GO TO 70
      GO TO 100
C
C---- CHECK FOR ENOUGH DATA PROCESSED
   90 IF(IDATA .LT. NPT) GO TO 80
C
C---- WRITE THE LAST RECORD TO DISC
```

```
PAGE 0003  DTRND  9:45 AM  MON.,  9  MAR., 1981
```

```
      IF(IPT .NE. 1) CALL WRITF(IDCB,IER,IBUF,128,IREC+1)
C
C---- CLOSE THE FILE
  100 CALL CLOSE(IDCB)
C
C---- REPORT THE RESULTS
      WRITE(LUTTY,1050) FIRST,FLAST,NPT,SUM,SLOPE,BIAS
 1050 FORMAT(" FILE DETRENDED"/
     *" FIRST=",F10.4," FLAST=",F10.4," NPT=",I5/
     *" AVERAGE=",F10.4," SLOPE=",F10.4," BIAS=",E10.4)
C
C---- PROCESS ANOTHER FILE
  110 WRITE(LUTTY,1060)
 1060 FORMAT(" DETREND ANOTHER FILE? (YE OR NO) _")
      READ(LUTTY,1010) I
      IF(I .EQ. 2HYE) GO TO 10
      STOP
      END


 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 01041      COMMON = 00000
```

```
0129          END$
```

```
FTN,L
      PROGRAM LSTAP,3,80
C
C---- PROGRAM TO LIST SELECTED RECORDS AND SUBRECORDS OF
C     GEOMAGNETIC VARIATION DATA TAPES.
C
C     WRITTEN BY D. V. FITTERMAN, U.S.G.S., FEBRUARY 1978
C     MODIFIED 24 DECEMBER 1980
C
      DIMENSION IAB(2),IHED(128),IBUF(1024)
      EQUIVALENCE (AB,IA,IAB(1)),(IB,IAB(2))
      EQUIVALENCE (NRATE,IHED(9)),(NCHAN,IHED(10)),(IBUFL,IHED(26)),
     *(NWORD,IHED(52)),(NSCAN,IHED(53))
      DATA LUTTY/1/,LUTAP/8/,LUPRT/6/
C
C---- IS TAPE POSITIONED PROPERLY?
      WRITE(LUTTY,1000)
 1000 FORMAT(" IS TAPE AT BEGINNING OF FILE? (YE OR NO) _")
      READ(LUTTY,1010) I
 1010 FORMAT(A2)
      IF(I .NE. 2HYE) STOP
C
C---- READ HEADER
      AB=EXEC(1,100B+LUTAP,IHED,128)
      NREC=1
C
C---- CHECK FOR EOF
      IF(IAND(IA,200B) .NE. 0) GO TO 100
      WRITE(LUTTY,1020) (IHED(I),I=1,23),(IHED(I),I=27,51)
 1020 FORMAT(//" VER=",I5," TRANSCRIPITION DAY=",I3," YEAR=",I4/
     *" TAPE FILE #",I4," LOC=",2A2, " CASS ID #",I2," INST. #",I2/
     *" SCAN RATE=",I1," CHAN/SCAN=",I1/
     *" RESET TIME=",I2,":",I2," DAY=",I2," MON=",I2," YR=",I4/
     *" OFF   TIME=",I2,":",I2," DAY=",I2," MON=",I2," YR=",I4,
     *" SW=",I2,":",I2,".",I1/1X,25A2)
C
C---- INPUT BOUNDS OF LISTING
   20 WRITE(LUTTY,1030)
 1030 FORMAT(/" START: IREC(>1)      ISREC(1-32)? _")
      READ(LUTTY,*) IREC,ISREC
      IF(IREC .LT. 1 .OR. ISREC .LT. 1 .OR. ISREC .GT. 32)
     *GO TO 20
   30 WRITE(LUTTY,1050)
 1050 FORMAT("  STOP: JREC(>=IREC)  ISREC(>=ISREC)? _")
      READ(LUTTY,*) JREC,JSREC
      IF(JREC .LT. IREC) GO TO 30
      IF(JSREC .GT. 32) GO TO 30
      MIN=1
      IF(JREC .EQ. IREC) MIN=ISREC
      IF(JSREC .LT. MIN) GO TO 30
   50 WRITE(LUTTY,1060)
 1060 FORMAT(" OUTPUT FORMAT: 1=HEADER, 2=HEADER + DATA? _")
      READ(LUTTY,*) IFORM
      IF(IFORM .NE.1 .AND. IFORM .NE. 2) GO TO 50
C
```

```
C---- LOCATE FIRST RECORD ON TAPE
      CALL LOCAT(LUTAP,NREC,IREC,IERR)
C
C---- CHECK FOR POSITIONING ERROR
      IF(IERR .EQ. 0) GO TO 55
      WRITE(LUTTY,1070) NREC,IREC
 1070 FORMAT(" EOF DURING POSITIONING, NREC=",I4," IREC=",I4)
      GO TO 110
   55 CALL EXEC(3,1100B+LUPRT,10)
      WRITE(LUPRT,1020) (IHED(I),I=1,23),(IHED(I),I=27,51)
      CALL EXEC(3,1100B+LUPRT,1)
C
C---- READ RECORD
   60 AB=EXEC(1,100B+LUTAP,IBUF,IBUFL)
C
C---- CHECK FOR EOF
      IF(IAND(200B,IA) .EQ. 200B) GO TO 110
   70 CALL OUT(IFORM,IBUF,LUPRT,NCHAN,NSCAN,NWORD,IREC,ISREC)
C
C---- ADVANCE SUBRECORD COUNTER
      ISREC=ISREC+1
C
C---- DONE?
      IF(ISREC .GT. JSREC .AND. IREC .EQ. JREC) GO TO 90
      ISREC=MOD(ISREC-1,32)+1
C
C---- MORE SUBRECORDS IN THIS RECORD?
      IF(ISREC .NE. 1) GO TO 70
C
C---- ADVANCE TO NEXT RECORD
      IREC=IREC+1
      GO TO 60
C
C---- UPDATE TAPE POSITION POINTER
   90 NREC=IREC+1
      GO TO 120
  100 WRITE(LUTTY,1080)
 1080 FORMAT(" EOF ON HEADER READ")
      GO TO 130
  110 WRITE(LUTTY,1090) IREC
 1090 FORMAT(" EOF WHILE READING RECORD",I5)
C
C---- BACKSPACE TWO FILE MARKS
      CALL EXEC(3,1400B+LUTAP)
      AB=EXEC(3,1400B+LUTAP)
C
C---- FORWARD SPACE OVER EOF AND HEADER
      IF(IAND(100B,IA) .NE. 100B) CALL EXEC(3,300B+LUTAP)
      CALL EXEC(3,300B+LUTAP)
      NREC=1
      WRITE(LUTTY,1100)
 1100 FORMAT(" TAPE POSITIONED AT BEGINNING FOR FIRST DATA RECORD")
C
C---- LIST ANOTHER SEGMENT
  120 WRITE(LUTTY,1110)
 1110 FORMAT(" LIST ANOTHER SEGMENT? (YE OR NO)? _")
      READ(LUTTY,1010) I
      IF(I .EQ. 2HYE) GO TO 20
C
C---- POSITION TAPE AT BEGINNING OF FILE
  130 CALL EXEC(3,1400B+LUTAP)
C
C---- CHECK TAPE STATUS
      AB=EXEC(3,600B+LUTAP)
      IF(IAND(IA,100B) .NE. 100B) CALL EXEC(3,300B+LUTAP)
      WRITE(LUTTY,1120)
 1120 FORMAT(" TAPE POSITIONED AT BEGINNING OF FILE")
      STOP
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 01987      COMMON = 00000
```

```
      SUBROUTINE LOCAT(LUTAP,NREC,IREC,IERR)
      DIMENSION IAB(2)
      EQUIVALENCE (IAB(1),IA,AB),(IAB(2),IB)
C
C---- RESET ERROR FLAG
      IERR=0
C
C---- CHECK FOR TAPE PROPERLY POSITIONED
      IF(IREC .EQ. NREC) RETURN
C
C---- DETERMINE DIRECTION TO MOVE TAPE
      IF(NREC .LT.IREC) GO TO 100
C
C---- BACKWARD SPACE TAPE ONE RECORD
   10 AB=EXEC(3,200B+LUTAP)
C
C---- CHECK FOR EOF
      IF(IAND(IA,200B) .EQ. 200B) GO TO 20
      NREC=NREC-1
C
C---- CORRECT POSITION?
      IF(NREC .EQ. IREC) RETURN
      GO TO 10
C
C---- EOF ENCOUNTERED
   20 IERR=1
      RETURN
C
C---- FORWARD SPACE TAPE ONE RECORD
  100 AB=EXEC(3,300B+LUTAP)
C
C---- CHECK FOR EOF
      IF(IAND(IA,200B) .EQ. 200B) GO TO 20
      NREC=NREC+1
C
C---- CORECT POSITION?
      IF(NREC .EQ. IREC) RETURN
      GO TO 100
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00091       COMMON = 00000
```

```
      SUBROUTINE OUT(IFORM,IBUF,LUPRT,NCHAN,NSCAN,NWORD,IREC,ISREC)
      DIMENSION IBUF(1)
C
C---- SET SUBRECORD POINTER
      IPTR=NWORD*(ISREC-1)
C
C---- COMPUTE CLOCK
      CLOCK=32768.*IBUF(IPTR+1)+IBUF(IPTR+2)
C
C---- OUTPUT HEADER
      WRITE(LUPRT,1000) IREC,ISREC,CLOCK,(IBUF(IPTR+I),I=3,7)
 1000 FORMAT(" IREC=",I4," ISREC=",I2," CLK=",F8.0," CASS=",I2,
     *" INST=",I3," NSCAN=",I2," NCHAN=",I1," DF=",@6,"B")
C
C---- CHECK FOR DATA LISTING
      IF(IFORM .EQ. 1) RETURN
C
C---- PRINT HEADER
      WRITE(LUPRT,1010) (I,I=1,NCHAN)
 1010 FORMAT(" SCAN  CHAN=  ",7(I1,4X))
C
C---- SET SCAN POINTER
      JPTR=7-NCHAN
      DO 10 I=1,NSCAN
      JPTR=JPTR+NCHAN
   10 WRITE(LUPRT,1020) (I,(IBUF(IPTR+JPTR+J),J=1,NCHAN))
 1020 FORMAT(2X,I2,7X,7(I5))
C
C---- ADVANCE PAPER
      CALL EXEC(3,1100B+LUPRT,1)
      RETURN
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00231      COMMON = 00000
 PAGE 0006 FIN.   9:45 AM  MON.,  9  MAR., 1981

0196          END$
```

```
 PAGE 0001  FTN.   9:45 AM  MON.,  9  MAR., 1981

FTN,L
      PROGRAM LSDSK,3,80
C
C---- PROGRAM TO LIST A FILE OF RECORDS OF A FILE CREATED
C     BY PROGRAM SLECT  OR ANY OTHER TYPE 1 FILES (128 WDS/REC).
C
C---- WRITTEN BY D.V. FITTERMAN, U.S.G.S., AUGUST 1976
C     MODIFIED 24 DECEMBER 1980
C
      DIMENSION IDCB(144),IFILE(3),IBUF(128)
      DATA LUTTY/1/,LUPRT/6/
C
C---- INPUT FILE NAME
   10 WRITE(LUTTY,1000)
 1000 FORMAT(" FILE NAME TO BE LISTED? _")
      READ(LUTTY,1010) IFILE
 1010 FORMAT(3A2)
C
C---- OPEN THE FILE
      CALL OPEN(IDCB,IER,IFILE,2)
      IF(IER .EQ. 1) GO TO 20
C
C---- OPENING ERROR
      IF(IER .LT. 0) WRITE(LUTTY,1020) IFILE,IER
 1020 FORMAT(" OPENING ERROR: FILE=",3A2," IER=",I5)
      GO TO 70
C
C---- GET FILE PARAMETERS - NUMBER OF RECORDS
   20 CALL LOCF(IDCB,IER,IREC,I,I,NSEC)
      NREC=NSEC/2
      WRITE(LUTTY,1030) NREC
 1030 FORMAT(" FILE HAS ",I5," RECORDS")
      WRITE(LUTTY,1040)
 1040 FORMAT(" LIST ENTIRE FILE? (YE OR NO) _")
      READ(LUTTY,1010) I
      IF(I .NE. 2HYE) GO TO 60
C
C---- LIST ENTIRE FILE
      DO 30 I=1,NREC
C
C---- READ A RECORD
      CALL READF(IDCB,IER,IBUF)
      IF(IER .GE. 0) GO TO 40
      WRITE(LUTTY,1050) IER,I
 1050 FORMAT(" READ ERROR=",I5," RECORD=",I5)
      GO TO 30
   40 CALL LIST(LUPRT,IFILE,I,IBUF)
   30 CONTINUE
C
C---- CLOSE FILE
   50 CALL CLOSE(IDCB)
   70 WRITE(LUTTY,1060)
 1060 FORMAT(" LIST ANOTHER FILE? (YE OR NO) _")
       READ(LUTTY,1010) I
      IF(I .EQ. 2HYE) GO TO 10
```

```
  999 STOP
C
C---- PRINT SPECIFIED RECORDS OF FILE ONLY
   60 WRITE(LUTTY,1070)
 1070 FORMAT(" RECORD TO BE LISTED? (.LE. 0 TO STOP) _")
      READ(LUTTY,*) N
C
C---- TEST FOR STOP
      IF(N .LE. 0) GO TO 50
C
C---- CHECK FOR RECORD NUMBER OUTSIDE LIMIT
      IF(N .GT. NREC) GO TO 60
C
C---- READ A RECORD
      CALL READF(IDCB,IER,IBUF,128,LEN,N)
      IF(IER .GE. 0) GO TO 80
      WRITE(LUTTY,1050) IER,N
      GO TO 60
C
C---- LIST THE RECORD
   80 CALL LIST(LUPRT,ITITLE,N,IBUF)
      GO TO 60
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00628      COMMON = 00000
```

```
      SUBROUTINE LIST(LUPRT,IFILE,I,IBUF)
      DIMENSION IBUF(128),IFILE(3)
C
C------ ADVANCE PRINTER 4 LINES
      CALL EXEC(3,1100B+LUPRT,2)
C
C---- WRITE FILE NAME AND RECORD NUMBER
      WRITE(LUPRT,1000) IFILE,I
 1000 FORMAT(" FILE=",3A2," RECORD=",I5,1X)
C
C---- SKIP A LINE
      CALL EXEC(3,1100B+LUPRT,1)
C
C---- PRINT THE RECORD
      N=-15
      DO 10 K=1,8
      N=N+16
   10 WRITE(LUPRT,1010) N,(IBUF(N+J-1),J=1,16)
 1010 FORMAT(" N=",I3,16I7)
      RETURN
      END
```

```
FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


**  NO WARNINGS **  NO ERRORS **   PROGRAM = 00111       COMMON = 00000
```

```
 PAGE 0004  FTN.   9:45 AM  MON.,  9  MAR., 1981

0100         END$
```

```
 PAGE 0001  FTN.   9:46 AM  MON.,  9  MAR., 1981

FTN,L
      PROGRAM PLOT0,3,80
C
C---- PROGRAM TO QUICKLY PLOT GEOMAGNETIC VARIATION DAILY
C     MAGNETOGRAMS WITHOUT HAVING TO SORT VECTORS.  RUNS
C     PLOTTER IN STRIP CHART MODE.  DOES NOT APPLY DOCUMENTATION.
C
C     WRITTEN BY D.V. FITTERMAN, U.S.G.S., MAY 1977
C     MODIFIED 9 MARCH 1981
C
      DIMENSION IAB(2),IHED(128),IBUF(1024),IRAS(132),G(7),FMIN(7),
     *FMAX(7),SCALE(7),MCLO(7),MCHI(7),MTICK(132),MEND(132),
     *MBORD(132),MHOUR(132),MASK(528)
      EQUIVALENCE (IAB(1),IA,AB),(IAB(2),IB),(NCHAN,IHED(10)),
     *(NTPWD,IHED(26)),
     *(NWORD,IHED(52)),(NSCAN,IHED(53)),(MEND(1),MASK(1)),
     *(MTICK(1),MASK(133)),(MBORD(1),MASK(265)),(MHOUR(1),MASK(397))
      DATA MCHI/1651,1426,1201,976,751,526,301/
      DATA MCLO/1451,1226,1001,776,551,326,101/
      DATA LUTTY/1/,LUPRT/6/,LUTAP/8/
      DATA G(6)/0.4882813/,G(7)/0.4882813/,DLINE/96.0/
C
C---- READ TAPE HEADER
   10 AB=EXEC(1,100B+LUTAP,IHED,128)
C
C---- CHECK FOR EOF
      IF(IAND(IA,200B) .NE. 0) STOP
C
C---- WRITE HEADER
      WRITE(LUTTY,1000) (IHED(I),I=1,23),(IHED(I),I=27,51)
 1000 FORMAT(//" VER=",I5," TRANSCRIPTION DAY=",I3," YEAR=",I4/
     *" TAPE FILE #",I4," LUC=",2A2," CASS ID #",I2," INST. #",I2/
     *" SCAN RATE=",I1," CHAN/SCAN=",I1/
     *" RESET TIME=",I2,":",I2," DAY=",I2," MON=",I2," YR=",I4/
     *" OFF   TIME=",I2,":",I2," DAY=",I2," MON=",I2," YR=",I4,
     *" SW=",I2,":",I2,".",I1/1X,25A2)
C
C---- COMPUTE CHANNEL GAINS
      DO 20 I=1,3
      G(I)=0.4882813
      IF(IHED(I+53) .NE. 0) G(I)=IHED(I+53)/2048.
   20 CONTINUE
      DO 30 I=1,2
      G(I+3)=1.0
      IF(IHED(I+56) .NE. 0 .AND. IHED(I+58) .NE. 0)
     *G(I+3)=4882.812/IHED(I+56)/IHED(I+58)
   30 CONTINUE
C
C---- OUTPUT MIN AND MAX, INPUT FMIN AND FMAX
      WRITE(LUTTY,1010)
 1010 FORMAT(/8X," ALLOWABLE",6X,"DESIRED"/
     *1X,"CHAN",3X,"MIN",4X,"MAX",4X,"MIN",4X,"MAX")
      DO 40 I=1,NCHAN
   50 FMIN(I)=-2048.*G(I)
      FMAX(I)=2047.*G(I)
      WRITE(LUTTY,1020) I,FMIN(I),FMAX(I)
 1020 FORMAT(3X,I1,2X,F6.0,1X,F6.0," _")
      READ(LUTTY,*) FMIN(I),FMAX(I)
      IF(FMIN(I) .GE. FMAX(I)) GO TO 50
   40 CONTINUE
C
C---- COMPUTE LOW AND HIGH COLUMN NUMBERS
C     USING VALUES IN ARRAYS MCLO AND MCHI GIVES 2" WIDE
C     PLOTS WITH 1/4" SPACING.
C     SEE DATA STATEMENTS
C
C---- COMPUTE BORDER WORD
      DO 60 I=1,528
   60 MASK(I)=0
C
C---- SET LINE LENGTH
      LENG=MCHI(8-NCHAN)
      LWORD=LENG/16
      IF(16*LWORD .LT. LENG) LWORD=LWORD+1
C
C---- LOOP OVER CHANNELS
      DO 70 I=1,NCHAN
      K=7-NCHAN+I
C
C---- END MASK
      DO 80 J=MCLO(K),MCHI(K)
   80 CALL INDOT(MEND,J)
C
C---- HOUR MASK
      DO 90 J=1,16
      CALL INDOT(MHOUR,MCLO(K)+J-1)
   90 CALL INDOT(MHOUR,MCHI(K)-J+1)
C
C---- BORDER MASK
      CALL INDOT(MBORD,MCLO(K))
      CALL INDOT(MBORD,MCHI(K))
C
C---- SCALE FACTORS  2" PER CHANNEL
      SCALF(I)=200./(FMAX(I)-FMIN(I))
C
C---- TICK MASK  TEN PER CHANNEL
      DTICK=(MCHI(K)-MCLO(K))/10.
      DO 100 J=1,11
      ITICK=FLOAT(J-1)*DTICK
  100 CALL INDOT(MTICK,MCLO(K)+ITICK)
   70 CONTINUE
C
C---- READ FIRST RECORD
      AB=EXEC(1,100B+LUTAP,IBUF,NTPWD)
C
C---- COMPUTE  CLOCK AT END OF CURRENT DAY
      CALL EODAY(IHED(11),IHED(12),CLKF,CLKI)
C
C---- COMPUTE CLOCK AT BEGINNING OF CURRENT DAY
      CLKI=CLKI-172800.
      DT=2**IHED(9)
C
C---- COMPUTE DAY OF YEAR
      NDAY=0
      CALL DAY(NDAY,IHED(13),IHED(14),IHED(15))
      NYEAR=IHED(15)
C
C---- SET POINTERS
      IEOD=0
      IEOF=0
      IPT=0
      ISREC=1
      JPT=7
      ISCAN=1
      CLK=32768.*IBUF(1)+IBUF(2)
      CLOCK=CLK-(NSCAN-1)*DT
      LINE=0
      OTIME=0.0
      CLINE=CLKI
C
C---- OUTPUT HEADER
  110 CALL EXEC(3,1100B+LUPRT,10)
      WRITE(LUPRT,1030) IHED(5),IHED(6),IHED(4),IHED(7),IHED(8),
     *IHED(9),DT,(IHED(I),I=27,51)
 1030 FORMAT(11X,2A2,"  TAPE FILE #",I3,"  CASSETTE #",I2,
     *"  INSTRUMENT #",I2,"  SCAN RATE=",I1,"  DTICK=",F4.1/
     *11X,25A2)
      WRITE(LUPRT,1035) NYEAR,NDAY
 1035 FORMAT(10X," YEAR=",I4," DAY=",I3)
      WRITE(LUPRT,1040)
 1040 FORMAT(10X," CHAN  MIN     MAX")
      WRITE(LUPRT,1050) (I,FMIN(I),FMAX(I),I=NCHAN,1,-1)
 1050 FORMAT(10X,7(1X,I1,": ",F7.1,1X,F7.1,1X))
      CALL EXEC(3,1100B+LUPRT,6)
C
C---- SELECT LINE
C
C---- CHECK FOR CENTRAL PART
  120 IF(LINE .GE. 15 .AND. LINE .LE. 1785) GO TO 130
C
C---- CHECK FOR FIRST OR LAST LINE
      IF(LINE .EQ. 1800 .OR. LINE .EQ. 0) GO TO 140
C
C---- TICK LINE
      MPT=132
      GO TO 200
C
C---- END LINE
  140 MPT=0
      GO TO 200
C
C---- CHECK FOR HOUR MARK
  130 IF(MOD(LINE,75) .NE. 0) GO TO 150
C
C---- HOUR LINE
```

```
      MPT=396
      GO TO 200
C
C---- BORDER LINE
  150 MPT=264
  200 DO 210 I=1,LWORD
  210 IRAS(I)=MASK(I+MPT)
C
C---- END OF DATA?
  230 IF(IEOD .GT. 0) GO TO 300
C
C---- DATA BEYOND CURRENT LINE?
      IF(CLOCK .GT. CLINE+DTIME) GO TO 300
C
C---- MERGE DATA
      DO 240 I=1,NCHAN
      F=G(I)*(IBUF(JPT+I)-2048)
      IF(F .GE. FMAX(I)) GO TO 240
      IF(F .LE. FMIN(I)) GO TO 240
      K=7-NCHAN+I
      ICOL=SCALE(I)*(F-FMIN(I))+MCOL(K)
      CALL INDOT(IRAS,ICOL)
  240 CONTINUE
C
C---- ADVANCE SCAN
      ISCAN=ISCAN+1
C
C---- CHECK FOR ENOUGH SCANS
      IF(ISCAN .GT. NSCAN) GO TO 250
      CLOCK=CLOCK+DT
      JPT=JPT+NCHAN
      GO TO 230
C
C---- RESET SCANS
  250 ISCAN=1
      ISREC=ISREC+1
C
C---- CHECK FOR ENOUGH SUBRECORDS
      IF(ISREC .GT. 32) GO TO 260
      IPT=IPT+NWORD
      JPT=IPT+7
C
C---- CHECK FOR END OF DATA
      IF(IBUF(IPT+4) .EQ. 0) IEOD=1
      CLKI=CLK
      CLK=32768.*IBUF(IPT+1)+IBUF(IPT+2)
      CLOCK=CLK-DT*(NSCAN-1)
      IF(CLK .GT. CLKI) GO TO 230
      GO TO 280
C
C---- READ TAPE RECORD
  260 AB=EXEC(1,100B+LUTAP,IBUF,NTPWD)
C
C---- CHECK FOR EOF
      IF(IAND(IA,200B) .NE. 0) GO TO 270
```

```
C
C---- RESET POINTERS
      IPT=0
      ISREC=1
      JPT=7
C
C---- CHECK FOR END OF DATA
      IF(IBUF(4) .EQ. 0) IEOD=1
      ISCAN=1
      CLKI=CLK
      CLK=32768.*IBUF(1)+IBUF(2)
      CLOCK=CLK-DT*(NSCAN-1)
      IF(CLK .GT. CLKI) GO TO 230
  280 IF(OTIME .EQ. 0.0) CLINE=CLINE-1048576.
      IF(OTIME .GT. 0.0) OTIME=0.0
      GO TO 230
C
C---- SET END OF DATA FLAG
  270 IEOD=1
      IEOF=1
      IF(LINE .EQ. 0) GO TO 310
      GO TO 230
C
C---- OUTPUT LINE
  300 CALL EXEC(2,100B+LUPRT,IRAS,LWORD)
C
C---- ADVANCE LINE AND CLOCK
      LINE=LINE+1
C
C---- OUTPUT ENOUGH LINES?
      IF(LINE .GE. 1801) GO TO 310
      CLINE=CLINE+DLINE
      IF(CLINE .LT. 1048576.) GO TO 120
      CLINE=CLINE-1048576.
      OTIME=1048576.
      GO TO 120
C
C---- DAY FINISHED, ADVANCE PAPER
  310 CALL EXEC(3,1100B+LUPRT,10)
C
C---- ANY MORE DATA?
      IF(IEOD .NE. 0) GO TO 400
C
C---- ADVANCE DAY
      CALL DAY(NDAY,IHED(13),IHED(14),NYEAR)
C
C---- RESET LINE
      LINE=0
      GO TO 110
C
C---- ADVANCE PAPER
  400 CALL EXEC(3,1100B+LUPRT,50)
C
C---- SKIP PAST EOF MARK IF NECESSARY
      CALL EXEC(3,1300B+LUTAP)
```

```
C
C---- PLOT NEXT FILE?
      WRITE(LUTTY,1060)
 1060 FORMAT(" PLOT NEXT FILE? (YE OR NO) _")
      READ(LUTTY,1070) I
 1070 FORMAT(A2)
      IF(I .EQ. 2HYE) GO TO 10
      STOP
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 03464      COMMON = 00000
```

```
      SUBROUTINE ENDAY(IHR,MIN,CLKI,CLKJ)
C
C---- SUBROUTINE TO COMPUTE CLOCK AT END OF DAY
      ITMIN=60-MIN
      ITHR=23-IHR
      IF(ITMIN .NE. 60) GO TO 10
      ITMIN=0
      ITHR=ITHR+1
   10 CLKJ=CLKI+7200.*ITHR+120.*ITMIN
      RETURN
      END


  FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00058      COMMON = 00000
```

```
      SUBROUTINE DAY(NDAY,IDAY,IMON,IYEAR)
C
C---- COMPUTES DAY OF YEAR FOR NDAY=0, OTHERWISE ADVANCES DAY OF YEAR
      DIMENSION JDAY(12)
      DATA JDAY/0,31,59,90,120,151,181,212,243,273,304,334/
      IL=0
C
C---- CHECK FOR LEAP YEAR
      IF(MOD(IYEAR,4) .NE. 0) GO TO 30
      IF(MOD(IYEAR,100) .NE. 0) GO TO 20
      IF(MOD(IYEAR,400) .NE. 0) GO TO 30
   20 IL=1
   30 IF(NDAY .NE. 0) GO TO 10
      NDAY=IDAY+JDAY(IMON)+IL
      RETURN
C
C---- ADVANCE DAY
   10 NDAY=NDAY+1
C
C---- CHECK FOR NEXT YEAR
      IF(NDAY .LT. 365+IL) RETURN
      NDAY=1
      IYEAR=IYEAR+1
      RETURN
      END
```

```
FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

**  NO WARNINGS **  NO ERRORS **   PROGRAM = 00093      COMMON = 00000
```

```
0321         END$
```

```
 PAGF 0001
0001                   ASMB,L,T,C
INDUT R 000002
.ENTR X 000001
RAS   R 000000
COL   R 000001
MASK  R 000022
MASKS R 000023
**  NO ERRORS PASS#1 **RTE ASMB 760924**
```

```
 PAGE 0002 #01
0001                  ASMB,L,T,C
0002  00000                 NAM INDOT,3,80
0003                        ENT INDOT
0004                        EXT .ENTR
0005*
0006*     ROUTINE TO PLACE A DOT AT A PARTICULAR BIT POSITION IN
0007*     ARRAY RAS.  THE FIRST WORD OF ARRAY RAS CONTAINS COLUMNS
0008*     1 THROUGH 16, THE I-TH WORD CONTAINS COLUMNS 16*(I-1)
0009*     THROUGH 16*(I-1)+16.
0010*
0011*     WRITTEN BY D. V. FITTERMAN, U.S.G.S., MAY 1977
0012*     MODIFIED 25 MAY 1977
0013*
0014  00000 000000  RAS   BSS 1         ADDR OF RASTER BUFFER
0015  00001 000000  COL   BSS 1         ADDR OF COLUMN TO BE SET
0016  00002 000000  INDOT NOP
0017  00003 016001X       JSB .ENTR     RESOLVE INDIRECT ADDRESSES
0018  00004 000000R       DEF RAS
0019  00005 162001R       LDA COL,I     LOAD COLUMN NUMBER
0020  00006 012044R       AND =B17      MAS OFF LOW ORDER BITS
0021  00007 042023R       ADA MASKS     ADD ADDRESS OF MASKS
0022  00010 072022R       STA MASK      SAVE MASK ADDRESS
0023  00011 007400        CCB           "B"=-1
0024  00012 146001R       ADB COL,I     ADD COLUMN VALUE
0025  00013 101044        LSR 4         DIVIDE BY 16
0026  00014 046000R       ADB RAS       ADD OFFSET TO RAS ADDRESS
0027  00015 076000R       STB RAS
0028  00016 105773        SBS MASK,I RAS,I
      00017 100022R
      00020 100000R
0029  00021 126002R       JMP INDOT,I
0030  00022 000000  MASK  BSS 1         ADDRESS OF MASK TO USE
0031  00023 000024R MASKS DEF *+1       ADDRESS OF FIRST MASK
0032  00024 000001        OCT 1
0033  00025 100000        OCT 100000
0034  00026 040000        OCT 40000
0035  00027 020000        OCT 20000
0036  00030 010000        OCT 10000
0037  00031 004000        OCT 4000
0038  00032 002000        OCT 2000
0039  00033 001000        OCT 1000
0040  00034 000400        OCT 400
0041  00035 000200        OCT 200
0042  00036 000100        OCT 100
0043  00037 000040        OCT 40
0044  00040 000020        OCT 20
0045  00041 000010        OCT 10
0046  00042 000004        OCT 4
0047  00043 000002        OCT 2
      00044 000017
0048                      END INDOT
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

```
PAGE 0003                  INDOT
                 CROSS-REFERENCE SYMBOL TABLE

.ENTR 00004   00017

=B17  .....   00020

COL   00015   00019   00024

INDOT 00016   00003   00029   00048

MASK  00030   00022   00028

MASKS 00031   00021

RAS   00014   00018   00026   00027   00028
```

PAGE 0001 FTN.   9:47 AM MON., 9 MAR., 1981

```
FTN,L
      PROGRAM PLOT3,3,80
C---- PROGRAM TO PLOT GEOMAGNETIC VARIATION ARRAY DISC SOURCE
C     FILES.  FILES CAN BE EITHER ORIGINAL (O), LOW PASSED (L),
C     OR DECIMATED (D) DATA.
C
C     WRITTEN BY D.V. FITTERMAN, U.S.G.S., SEPTEMBER 1976
C     MODIFIED 6 JANUARY 1981
C
      COMMON IVEC(256),IVECS(256),NDCB(144),INVEC(4),NPTR,NREC
      DIMENSION NFILE(3),IFILE(3),IDCB(144),IBUF(128),IHED(128),
     *ITITL(25),JTITL(25),MERGE(3),IPLOT(3)
      EQUIVALENCE (IDCB(17),IBUF(1)),(NPT,IHED(66)),(FNPT,IHED(127))
      DATA LUTTY/1/,NFILE/2HVE,2HCT,2HRS/,MERGE/2HME,2HRG,2HE /,
     *IPLOT/2HPL,2HOT,2H  /
C
C---- DETERMINE FILE TO BE PLOTTED
   10 WRITE(LUTTY,1000)
 1000 FORMAT(//" FILE TO BE PLOTTED? (6 CHAR) _")
      READ(LUTTY,1010) IFILE
 1010 FORMAT(3A2)
C
C---- DETERMINE IF FILE EXISTS
      CALL OPEN(IDCB,IER,IFILE,2)
      IF(IER .GE. 0) GO TO 20
      WRITE(LUTTY,1020) IFILE,IER
 1020 FORMAT(" OPENING ERROR: FILE=",3A2," IER=",I5)
      GO TO 60
C
C---- FILE EXISTS READ FIRST RECORD AND OUTPUT HEADER
   20 CALL READF(IDCB,IER,IHED)
C
C---- CHECK FOR FILE LENGTH > 32767
      IF(NPT .GT. 0) GO TO 25
      WRITE(LUTTY,1025) FNPT
 1025 FORMAT(" FILE LENGTH=",F7.0," > 32767")
      CALL CLOSE(IDCB)
      GO TO 60
C
C---- COMPUTE TIME BETWEEN DATA POINTS
   25 DT=IHED(67)*IHED(68)/2.
      WRITE(LUTTY,1030) (IHED(I),I=4,9),(IHED(I),I=27,51),
     *(IHED(I),I=61,63),IHED(65),IHED(64),NPT,IHED(67),IHED(68),DT
 1030 FORMAT(" TAPE FILE #",I5," LOC=",2A2," CASS ID #",I3,
     *" INST. #",I3," SCAN RATE=",I1/1X,25A2/
     *" START OF DATA SEGMENT=",I2,":",I2,":",I2,1X,I4,".",I3/
     *" NPT=",I5," NDEC=",I5," ORIGINAL SI=",I3," TICKS  DT=",
     *F6.1," SEC")
C
C---- DETERMINE CHANNEL FROM FILE NAME
      CALL CHANL(IFILE,LUTTY,ICHAN)
C
C---- DETERMINE MINIMUM AND MAXIMUM VALUE
      WRITE(LUTTY,1040)
 1040 FORMAT(" SEARCH FOR MINIMUM AND MAXIMUM VALUE? (YE OR NO) _")
```

```
      READ(LUTTY,1010) I
      IF(I .EQ. 2HNO) GO TO 30
      CALL MINMX(IDCB,IBUF,NPT,MIN,MAX)
C
C---- COMPUTE VERTICAL SCALE FACTOR (UNITS/COUNT)
      CALL SCALE(IHED,ICHAN,VSCAL)
      FMAX=VSCAL*(MAX-2046)
      FMIN=VSCAL*(MIN-2048)
C
C---- IF VSCAL < 0 INTERCHANGE FMAX AND FMIN
      IF(VSCAL .GE. 0) GO TO 40
      T=FMAX
      FMAX=FMIN
      FMIN=T
   40 WRITE(LUTTY,1050) MAX,MIN,FMAX,FMIN
 1050 FORMAT(" DATA SET SEARCHED"/" MAX=",I5," MIN=",I5," FMAX=",
     *F9.2," FMIN=",F9.2/
     *" ANY CHANGES? (YE OR NO) _")
      READ(LUTTY,1010) I
      IF( I .EQ. 2HNO) GO TO 50
      WRITE(LUTTY,1060)
 1060 FORMAT(" INPUT  FMAX FMIN")
      READ(LUTTY,*) FMAX,FMIN
      GO TO 50
C
C---- COMPUTE VSCAL
   30 CALL SCALE(IHED,ICHAN,VSCAL)
      WRITE(LUTTY,1060)
      READ(LUTTY,*) FMAX,FMIN
C
C---- INPUT VERITCAL AND HORIZONTAL SIZE
   50 WRITE(LUTTY,1080)
 1080 FORMAT(" PLOT SIZE? (INCHES)"/" VERT  HORZ")
      READ(LUTTY,*) FVERT,FHORZ
      WRITE(LUTTY,1090)
 1090 FORMAT(" VERTICAL TICK INTERVAL? (UNITS) _")
      READ(LUTTY,*) VTICK
      WRITE(LUTTY,1100)
 1100 FORMAT(" HORIZONTAL TICK INTERVAL? (SECONDS) _")
      READ(LUTTY,*) HTICK
C
C---- INPUT TITLE AND SUBTITLE
      WRITE(LUTTY,1110)
 1110 FORMAT(" TITLE? (.LE. 50 CHAR) _")
      READ(LUTTY,1120) (ITITL(I),I=1,25)
 1120 FORMAT(25A2)
      WRITE(LUTTY,1130)
 1130 FORMAT(" SUBTITLE? (.LE. 50 CHAR) _")
      READ(LUTTY,1120) (JTITL(I),I=1,25)
C
C---- MORE THAN ONE COPY OF PLOT?
      WRITE(LUTTY,1140)
 1140 FORMAT(" MORE THAN ONE COPY OF THE PLOT? (YE OR NO) _")
      READ(LUTTY,1010) I
      IPARM=0
      IF(I .EQ. 2HYE) IPARM=-1
C
C---- FACTORS FOR CONVERTING COUNTS AND SECONDS TO STYLI
      VSTY=100.*FVERT*VSCAL/(FMAX-FMIN)
      HSTY=100.*FHORZ/(NPT*DT)
C
C---- CREATE VECTOR FILE
      CALL START(NFILE)
C
C---- PLOT BOX, TICK MARKS, AXIS LABELS, AND TITLES.  UPPER
C     LEFT HAND CORNER OF BOX LOCATED AT (IN0,100), I.E., ONE
C     INCH IN FROM LEFT MARGIN.
      IN0=MIN1(30000.,32700.+FMAX*VSTY/VSCAL)
      CALL BOX(IN0,100,FVERT,FHORZ,FMAX,FMIN,VTICK,HTICK,VSCAL,
     *VSTY,HSTY,NPT,DT,ITITL,JTITL,IFILE)
      IX0=IN0-FMAX*VSTY/VSCAL
C
C---- PLOT THE DATA
      CALL DATA(IX0,100,DT,NPT,VSTY,HSTY,VSCAL,FMAX,FMIN,
     *IDCB,IBUF)
C
C---- CLOSE THE INPUT FILE
      CALL CLOSE(IDCB)
C
C---- CLOSE THE VECTOR FILE
      CALL STOP
C
C---- MERGE THE SORTED GROUP OF VECTORS
      CALL EXEC(9,MERGE,NFILE,NFILE(2),NFILE(3),-10,0)
C
C---- PLOT THE SORTED VECTORS
      CALL EXEC(9,IPLOT,NFILE,NFILE(2),NFILE(3),-10,IPARM)
C
C---- CHECK FOR ANOTHER FILE TO PLOT
   60 WRITE(LUTTY,1150)
 1150 FORMAT(" PLOT ANOTHER FILE? (YE OR NO) _")
      READ(LUTTY,1010) I
      IF(I .EQ. 2HYE) GO TO 10
      STOP
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 01352      COMMON = 00662
```

```
      SUBROUTINE CHANL(IFILE,LUTTY,ICHAN)
C
C---- DETERMINES THE CHANNEL NUMBER FROM THE LAST CHARACTER
C     IN THE FILE NAME.  (X=1,Y=2,Z=3,E=4,F=5,ALL OTHERS=6)
      DIMENSION IFILE(3),ICHAR(6)
      DATA ICHAR/130B,131B,132B,105B,106B/
      LCHAR=IAND(IFILE(3),377B)
      DO 10 I=1,5
      IF(LCHAR .NE. ICHAR(I)) GO TO 10
      ICHAN=I
      RETURN
   10 CONTINUE
C
C---- UNABLE TO DO AUTOMATIC COMPONENT DETERMINATION
      WRITE(LUTTY,1000)
 1000 FORMAT(" UNABLE TO DETERMINE COMPONENT")
   20 WRITE(LUTTY,1010)
 1010 FORMAT(" PLEASE SPECIFY (1=X, 2=Y, 3=Z, 4=E, 5=F) _")
      READ(LUTTY,*) ICHAN
      IF(ICHAN .LT. 1 .OR. ICHAN .GT. 5) GO TO 20
      RETURN
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00122      COMMON = 00000
```

```
      SUBROUTINE MINMX(IDCB,IBUF,NPT,MIN,MAX)
C
C---- SUBROUTINE TO FIND MINIMUM AND MAXIMUM VALUES IN A FILE
      DIMENSION IDCB(144),IBUF(128)
C
C---- READ FIRST DATA RECORD
      CALL READF(IDCB,IER,IBUF)
C
C---- SET INITIAL VALUES FOR MIN AND MAX
      MIN=IBUF(1)
      MAX=IBUF(1)
      IPT=0
C
C---- START SEARCH LOOP
   20 DO 10 I=1,128
      IPT=IPT+1
      IF(IBUF(I) .LT. MIN) MIN=IBUF(I)
      IF(IBUF(I) .GT. MAX) MAX=IBUF(I)
      IF(IPT .GE. NPT) GO TO 30
   10 CONTINUE
      CALL READF(IDCB,IER,IBUF)
      GO TO 20
C
C---- REPOSITION ON SECOND RECORD OF FILE
   30 CALL READF(IDCB,IER,IBUF,128,1,1)
      RETURN
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00095      COMMON = 00000
```

```
      SUBROUTINE SCALE(IHED,ICHAN,VSCAL)
C
C---- DETERMINED SCALE FACTORS FOR CONVERTING FROM COUNTS TO UNITS
      DIMENSION IHED(128)
      IF(ICHAN .LT. 1 .OR. ICHAN .GT. 3) GO TO 10
C
C---- HX, HY, HZ
      VSCAL=IHED(ICHAN+53)/2048.
C
C---- TEST FOR OLD FILES WITH NO GAIN VALUE
C     USE VALUE OF 1000 NANOTESLA/2048 COUNTS
      IF(VSCAL .EQ. 0.0) GO TO 20
      RETURN
C
C---- EX, EY
   10 IF(ICHAN .LT. 4 .OR. ICHAN .GT. 5) GO TO 20
      VSCAL=4882.813/IHED(ICHAN+53)/IHED(ICHAN+55)
      RETURN
C
C---- OTHER CHANNELS AND OLD UNCALIBRATED MAGNETICS
   20 VSCAL=0.4882813
      RETURN
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00107      COMMON = 00000
```

```
      SUBROUTINE START(NFILE)
C
C---- CREAT VECTOR FILE
      COMMON IVEC(256),IVECS(256),NDCB(144),INVEC(4),NPTR,NREC
      DIMENSION NSIZE(2),NFILE(3)
      DATA NSIZE/-1,256/
      CALL CREAT(NDCB,IER,NFILE,NSIZE,2,0,-10)
      IF(IER .GE. 0) GO TO 10
      WRITE(1,1000) NFILE,IER
 1000 FORMAT(" CREATION ERROR: NFILE=",3A2," IER=",I5)
      STOP
   10 NPTR=0
      NREC=1
      RETURN
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00068      COMMON = 00662
```

```
      SUBROUTINE BOX(IX,IY,FX,FY,FMAX,FMIN,VTICK,HTICK,
     *VSCAL,VSTY,HSTY,NPT,DT,ITITL,JTITL,IFILE)
C
C---- PLOT BOX, AXIS TICKS, AXIS LABELS, AND TITLES
      COMMON IVEC(256),IVECS(256),NDCB(144),INVEC(4),NPTR,NREC
      DIMENSION ITITL(25),JTITL(25),IFILE(3),LABEL(3),IASCI(6)
      DATA LABEL/2HFI,2HLE,2H: /
C
C---- CONVERT INCHES TO STYLI
      IFX=100*FX
      IFY=100*FY
C
C---- LEFT SIDE
      INVEC(1)=IX
      INVEC(3)=IX-IFX
      INVEC(2)=IY
      INVEC(4)=IY
      CALL INSV(INVEC)
C
C---- LOWER SIDE
      INVEC(1)=INVEC(3)
      INVEC(2)=IY+IFY
      CALL INSV(INVEC)
C
C---- RIGHT SIDE
      INVEC(3)=IX
      INVEC(4)=INVEC(2)
      CALL INSV(INVEC)
C
C---- UPPER SIDE
      INVEC(1)=IX
      INVEC(2)=IY
      CALL INSV(INVEC)
C
C---- VERTICAL AXIS TICKS
C---- LOCATE ZERO LEVEL(STYLI)
      FL=FMAX*VSTY/VSCAL
      IX0=IX-FL
C
C---- LOCATE MINIMUM LEVEL (STYLI)
      FL=FMIN*VSTY/VSCAL
      IXMN=IX0+FL
      N=FMIN/VTICK
      N=N-1
      F=N*VTICK
      DXJ=VTICK*VSTY/VSCAL
      XJ=IX0+N*DXJ
      IFLAG=0
C
C---- BEGIN VERTICAL TICK LOOP
   10 IF(F .LT. FMIN) GO TO 20
      IF(IFLAG .EQ. 1) GO TO 30
      IFLAG=1
      FL=F
      IXL=XJ
   30 IF(F .GT. FMAX) GO TO 40
C
C---- PLOT VERTICAL AXIS TICKS
C     LEFT SIDE TICKS
      INVEC(1)=XJ
      INVEC(3)=XJ
      INVEC(2)=IY
      INVEC(4)=IY+15
      CALL INSV(INVEC)
C
C---- RIGHT SIDE TICKS
      INVEC(2)=INVEC(2)+IFY
      INVEC(4)=INVEC(2)-15
      CALL INSV(INVEC)
   20 XJ=XJ+DXJ
      F=F+VTICK
      GO TO 10
C
C---- LABEL UPPER LIMIT TICK MARK
   40 F=F-VTICK
      XJ=XJ-DXJ
      JX=XJ
      CALL CODE
      WRITE(IASCI,1000) F
 1000 FORMAT(F7.1)
      CALL CHAR(JX+7,IY-98,IASCI,7)
C
C---- LABEL LOWER LIMIT TICK MARK
      CALL CODE
      WRITE(IASCI,1000) FL
      CALL CHAR(JXL+7,IY-98,IASCI,7)
C
C---- CHECK FOR LABELING OF ZERO POINT
      IF(IXO .GT. JXL .AND. IXO .LT. IX) CALL CHAR(IXO+7,IY-21,1HO,1)
C
C---- HORIZONTAL AXIS TICKS
C     LOCATE ZERO
      IXO=IX-15
      IXMN=IX-IFX
      IXMX=IXMN+15
      IXL=IXMN-24
      DYI=HTICK*HSTY
      YJ=IY-DYI
      N=NPT*(DT/HTICK)+1
      TICK=-HTICK
C
C---- BEGIN HORIZONTAL TICK LOOP
      DO 50 I=1,N
      YJ=YJ+DYI
      TICK=TICK+HTICK
C
C---- UPPER SIDE TICKS
      INVEC(1)=IX
      INVEC(3)=IXO
      INVEC(2)=YJ
      INVEC(4)=YJ
      CALL INSV(INVEC)
C
C---- LOWER SIDE TICKS
      INVEC(1)=IXMX
      INVEC(3)=IXMN
      CALL INSV(INVEC)
C
C---- LABEL TICK MARKS
      JY=YJ
      CALL CODE
      WRITE(IASCI,1010) TICK
 1010 FORMAT(F6.0)
   50 CALL CHAR(IXL,JY-49,IASCI,6)
C
C---- TITLE AND SUBTITLE
      CALL CHAR(IX+42,IY,ITITL,50)
      CALL CHAR(IX+21,IY,JTITL,50)
C
C---- LABEL FILE NAME
      CALL CHAR(IX+63,IY,LABFL(1),6)
      CALL CHAR(IX+63,IY+84,IFILE,6)
      RETURN
      END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00586      COMMON = 00662
```

```
      SUBROUTINE DATA(IX0,IY0,DT,NPT,VSTY,HSTY,VSCAL,FMAX,FMIN,
     *IDCB,IBUF)
C
C---- SUBROUTINE TO PLOT DATA
      COMMON IVEC(256),IVECS(256),NDCB(144),INVEC(4),NPTR,NREC
      DIMENSION IDCB(1),IBUF(1)
C
C---- COMPUTE LIMITS IN STYLI
C     IX0 CORRESPONDS TO ZERO
      T=FMAX*VSTY/VSCAL
      IFMX=IX0+T
      T=FMIN*VSTY/VSCAL
      IFMN=IX0+T
C
C---- COMPUTE HORIZONTAL INCREMENT
      DYJ=DT*HSTY
C
C---- INITTIALIZE HORIZONTAL POINTER
      YJ=IY0
C
C---- READ FIRST RECORD
      CALL READF(IDCB,IER,IBUF)
      KPT=VSTY*(IBUF(1)-2048)+IX0
C
C---- CHECK LIMITS
      IF(KPT .GT. IFMX) KPT=IFMX
      IF(KPT .LT. IFMN) KPT=IFMN
      INVEC(3)=KPT
      INVEC(4)=YJ
      N=1
      IPT=2
C
C---- LOAD NEXT POINT
   10 KPT=VSTY*(IBUF(IPT)-2048)+IX0
      IF(KPT .GT. IFMX) KPT=IFMX
      IF(KPT .LT. IFMN) KPT=IFMN
      INVEC(1)=KPT
      YJ=YJ+DYJ
      INVEC(2)=YJ
      CALL INSV(INVEC)
      N=N+1
C
C---- CHECK FOR ENOUGH POINTS
      IF(N .GE. NPT) RETURN
C
C---- SHUFFLE VECTORS
      INVEC(3)=INVEC(1)
      INVEC(4)=INVEC(2)
C
C---- CHECK FOR EMPTY INPUT BUFFER
      IF(IPT .GE. 128) GO TO 20
      IPT=IPT+1
      GO TO 10
C
C---- READ ANOTHER RECORD
```

```
   20 CALL READF(IDCB,IER,IBUF)
      IPT=1
      GO TO 10
      END


 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00222       COMMON = 00662
```

```
      SUBROUTINE STOP
      COMMON IVEC(256),IVECS(256),NDCB(144),INVEC(4),NPTR,NREC
C
C---- PUT EOF MARK ON VECTOR FILE
      DO 10 I=1,4
   10 INVEC(I)=0
      DO 20 I=NPTR,253,4
   20 CALL INSV(INVEC)
C
C---- CLOSE VECTOR FILE RELEASING UNUSED DISC AREA
      CALL LOCF(NDCB,IER,I,I,1,KSEC)
      I=KSEC/2-2*(NREC-1)
      CALL CLOSE(NDCB,IER,I)
      RETURN
      END


 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00071      COMMON = 00662
```

```
      SUBROUTINE CHAR(IX,IY,ITEXT,LCHAR)
      COMMON IVEC(256),IVECS(256),NDCB(144),INVEC(4),NPTR,NREC
      DIMENSION ITEXT(1)
      N=1
      M=1
      INVEC(1)=IX
      INVEC(2)=IY
      INVEC(3)=32764
C
C---- LEFT BYTE
   10 INVEC(4)=IAND(ITEXT(M),77400B)/256+1000B
      CALL INSV(INVEC)
      IF(N .GE. LCHAR) RETURN
C
C---- HORIZONTAL SPACE
      N=N+1
      INVEC(2)=INVEC(2)+14
C
C---- RIGHT BYTE
      INVEC(4)=IAND(ITEXT(M),377B)+1000B
      CALL INSV(INVEC)
      IF(N .GE. LCHAR) RETURN
      M=M+1
C
C---- HORIZONTAL SPACE
      N=N+1
      INVEC(2)=INVEC(2)+14
      GO TO 10
      END
```

```
FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


**  NO WARNINGS **  NO ERRORS **   PROGRAM = 00114      COMMON = 00662
```

```
      SUBROUTINE INSV(JNVEC)
      COMMON IVEC(256),IVECS(256),NDCB(144),INVEC(4),NPTR,NREC
      DIMENSION VEC(128),VECS(128)
      DIMENSION JNVEC(4),IC(64),LP(10)
      EQUIVALENCE (VEC,IVEC),(VECS,IVECS)
C
C---- CHECK FOR ORDER OF FIRST COORDINATES
      II=0
C
C---- CHECK FOR CHARACTER
      IF(JNVEC(3) .EQ. 32764) GO TO 10
      IF(JNVEC(1) .LT. JNVEC(3)) II=2
C---- INSERT VECTOR
   10 DO 20 I=1,2
      IVEC(NPTR+I)=JNVEC(I+II)
   20 IVEC(NPTR+I+2)=JNVEC(I-II+2)
      NPTR=NPTR+4
      IF(NPTR .LE. 253) RETURN
      NPTR=0
C
C---- SORT VECTORS AND WRITE ON DISC
      CALL SORT(IC,VEC,VECS,64,LP,10)
      CALL WRITF(NDCB,IER,IVECS,0,NREC)
      IF(IER .GE.0) GO TO 30
      WRITE(1,1000) IER
 1000 FORMAT(" INSV: WRITE ERROR #",I3)
      PAUSE 30
C
C---- INCREMENT RECORD NUMBER
   30 NREC=NREC+1
      RETURN
      END
```

```
FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


**  NO WARNINGS **  NO ERRORS **   PROGRAM = 00215      COMMON = 00662
```

```
       SUBROUTINE SORT(IA,A,AS,N,PUSH,LPUSH)
       DIMENSION A(1),IA(1),AS(1)
       INTEGER PUSH(1),U,U1
C
       IF (N-1) 112,112,99
99     U=1
       DO 100 L=1,N
       IA(L)=U
100    U=U+4
       IF (N-1) 109,109,101
101    J=LPUSH-2
       M=0
       L1=1
       U1=N
102    IF (U1-L1) 107,107,103
103    K=KSORT(IA,A,L1,U1,L,U)
       IF (K) 107,107,104
104    IF (M-J) 106,106,105
105    STOP 6
106    M=M+2
       PUSH(M-1)=L
       PUSH(M)=U
       GO TO 102
107    IF (M) 109,109,108
108    L1=PUSH(M-1)
       U1=PUSH(M)
       M=M-2
       GO TO 102
109    L=1
       DO 111 J=1,N
       U=(IA(J)+1)/2
       AS(L)=A(U)
       AS(L+1)=A(U+1)
111    L=L+2
112    RETURN
       END
```

```
FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


**  NO WARNINGS **  NO ERRORS **   PROGRAM = 00187       COMMON = 00000
```

```
          FUNCTION KSORT(A,FA,L1,U1,L,U)
          INTEGER A(1),U1,U,T,X,P,Q,FA(1)
    C
          IF (U1-L1-1) 100,102,104
    100   KSORT=0
    101   CONTINUE
          RETURN
    102   IF (FA(A(L1))-FA(A(U1))) 103,100,100
    103   X=A(L1)
          A(L1)=A(U1)
          A(U1)=X
          GO TO 100
    104   KSORT=1
          P=(L1+U1)/2
          T=A(P)
          A(P)=A(L1)
          Q=U1
          K=L1
    106   K=K+1
          IF (K-Q) 107,107,113
    107   IF (FA(A(K))-FA(T)) 108,106,106
    108   IF (Q-K) 113,109,109
    109   IF (FA(A(Q))-FA(T)) 110,110,111
    110   Q=Q-1
          GO TO 108
    111   X=A(K)
          A(K)=A(Q)
          A(Q)=X
          Q=Q-1
          GO TO 106
    113   A(L1)=A(Q)
          A(Q)=T
          IF ((Q+Q)-(L1+U1)) 116,116,115
    115   L=L1
          U=Q-1
          L1=Q+1
          GO TO 101
    116   L=Q+1
          U=U1
          U1=Q-1
          GO TO 101
          END
```

```
 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 00230       COMMON = 00000
```

```
0585          END$
```

```
FTN4,L
      PROGRAM MERGE,3,80
C
C---- CHECKED 18 AUGUST 1976
      DIMENSION IDCBI(272),IDCBO(272),DO(128),AIN(512),NAME(6),
     1 IN(1024),JA(4),JAS(4),IAS(4),IA(4),NA(4),NAL(4),KDAT(5)
      DIMENSION JAL(4)
      EQUIVALENCE (IN(1),AIN(1)),(IA(1),IA1),(IA(2),IA2),(IA(3),IA3),
     1 (IA(4),IA4)
      EQUIVALENCE (KDAT(5),IDFLG)
      DATA JAS,IAS/1,129,257,385,1,257,513,769/
      DATA JAL/127,255,383,511/
      DATA IDEGT/77777B/
      DATA NBUF,NAME/4,3*0,2HME,2HRG,2HEP/
C
      CALL RMPAR(KDAT)
      DO 3 I=1,3
3     NAME(I)=KDAT(I)
      CALL OPEN(IDCBI,IER,NAME,0,0,KDAT(4),272)
      IF (IER) 4,5
4     STOP 10
5     CALL LOCF(IDCBI,IER,I,I,I,NREC)
      IN(1)=NREC/2
      NREC=NREC/4
      IF (NREC-4) 1000,6
6     IN (2)=256
6661  CALL PURGE(IDCBO,IER,NAME(4),0,KDAT(4))
      CALL CREAT(IDCBO,IER,NAME(4),IN,2,0,KDAT(4),272)
      IF (IER) 7,8
7     IF (KDAT(4)) 771,777,771
771   KDAT(4)=0
      GO TO 6661
777   WRITE(1,9971) IER,IN(1)
9971  FORMAT(" FMGR:",I5,", NO. BLOCKS:",I6)
      STOP 11
8     ICY1=1
      ICY2=4
      IPHI=1
      GO TO 60
C
C  NEST PHASE POINT, BUMP RECORDS/PHASE
50    IPHI=IPHI*4
C
C  IF IPHI GREATER THAN OR EQUAL TO # RECORDS
C THEN DONE..
      IF (IPHI-NREC) 55,300
55    CALL CLOSE(IDCBI)
      CALL CLOSE(IDCBO)
      I=ICY2
      ICY2=ICY1
      ICY1=I
C
C  STARTING ENTRY AT 105
      CALL OPEN(IDCBO,IER,NAME(ICY2),0,0,KDAT(4),272)
      CALL OPEN(IDCBI,IER,NAME(ICY1),0,0,KDAT(4),272)
```

```
60     NA(1)=1
       GO TO 105
100    NA(1)=NAL(NBUF)+1
       IF (NA(1)-NREC) 105,105,50
105    IA1=1
       DO 110 I=2,NBUF
       NA(I)=NA(I-1)+IPHI
       IF (NA(I)-NREC) 108,108,109
108    IA(I)=IAS(I)
       NAL(I-1)=NA(I)-1
       GO TO 110
109    NAL(I-1)=NREC
       NA(I)=NREC
       IA(I)=-1
110    CONTINUE
       NAL(NBUF)=MIN0(NAL(NBUF-1)+IPHI,NREC)
C
C  GET FIRST RECORD OF EACH BUFFER
       DO 120 I=1,NBUF
       IF (IA(I)) 122,115
115    CALL READF(IDCBI,IER,IN(IA(I)),256,L,NA(I))
120    JA(I)=JAS(I)
122    CONTINUE
       IO=1
C
C  THIS SECTION SCANS FOR INITIAL NON-EMPTY
C  INPUT BUFFER
C  NOTE.. "DO" LOOPING COULD BE USED,  BUT IN THE INTERESTS
C  OF EFFICIENCY......
C
200    IF (IA1) 202,201
201    LWIN=IA1
       LWINF=1
       GO TO 220
202    IF (IA2) 204,203
203    LWIN=IA2
       LWINF=2
       GO TO 224
204    IF (IA3) 206,205
205    LWIN=IA3
       LWINF=3
       GO TO 228
206    IF (IA4) 100,230
C
C WHEN WE GET TO THIS POINT ALL INPUTS ARE EMPTY
C  ALSO,  BY DEFAULT OUTPUT ALSO SPILLED.
C
C  THIS SECTION DETERMIMES WINNER AMONG
C  REMAINING NON-EMPTY BUFFERS
220    IF (IA2) 224,221
221    IF (IN(LWIN)-IN(IA2)) 222,224
222    LWIN=IA2
       LWINF=2
224    IF (IA3) 228,225
225    IF (IN(LWIN)-IN(IA3)) 226,228
```

```
226   LWIN=IA3
      LWINF=3
228   IF (IA4) 240,229
229   IF (IN(LWIN)-IN(IA4)) 230,240
230   LWINF=4
C
C  WE NOW HAVE WINNER
C  PUT IN OUTPUT ARRAY AND WRITE IF FULL
240   I=JA(LWINF)
      DO(IO)=AIN(I)
      DO (IO+1)=AIN(I+1)
      IF (IO-127) 245,242
242   CALL WRITF(IDCBO,IER,DO)
      IO=1
      GO TO 250
245   IO=IO+2
C
C  THIS SECTION UPDATES WINNER INPUT BUFFER POINTSEES
C  AND INPUT NEW BLOCK IF REQUIRED.
250   IF (JA(LWINF)-JAL(LWINF)) 270,252
252   IF (NA(LWINF)-NAL(LWINF)) 256,254
254   IA(LWINF)=-1
      GO TO 200
256   NA(LWINF)=NA(LWINF)+1
      CALL READF(IDCBI,IER,IN(IAS(LWINF)),256,L,NA(LWINF))
      JA(LWINF)=JAS(LWINF)
      IA(LWINF)=IAS(LWINF)
      GO TO 200
270   JA(LWINF)=JA(LWINF)+2
      IA(LWINF)=IA(LWINF)+4
      GO TO 200
C
C  END OF SORT AFTER LAST MERGE SYSLCE
300   IF (ICY1-1) 301,301,302
301   CALL PURGE(IDCBI,IER,NAME(1),0,KDAT(4))
      CALL NAMF(IDCBO,IER,NAME(4),NAME(1),0,KDAT(4))
      GO TO 310
302   CALL PURGE(IDCBI,IER,NAME(4))
      CALL CLOSE(IDCBO)
310   CONTINUE
C
C  CHANGE OLDEND-FILE FLAGS TO REQUIRED
      CALL OPEN(IDCBI,IER,NAME,0,0,KDAT(4),272)
1000  CONTINUE
      CALL READF(IDCBI,IER,IN,256,L,NREC)
      DO 1010 I=1,256
      IF (IN(I)-IDFLG) 1010,1005,1010
1005  IN(I)=IDFGT
1010  CONTINUE
      CALL WRITF(IDCBI,IER,IN,256,NREC)
      CALL CLOSE(IDCBI)
      END
```

```
PAGE 0004  MERGE  9:48 AM  MON.,  9  MAR., 1981

 FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)

 **  NO WARNINGS **  NO ERRORS **   PROGRAM = 02566       COMMON = 00000
 PAGF 0005  FTN.   9:48 AM  MUN.,  9  MAR., 1981

0163         END$
```

```
 PAGE 0001  FTN.   9:49 AM  MON.,  9  MAR., 1981


FTN,L
      PROGRAM PLOT,3,80
C
C---- PROGRAM TO RASTERIZE AND SORT PLOTTED VECTORS
C
C     WRITTEN BY D. V. FITTERMAN, DECEMBER 1980
C     MODIFIED 6 JANUARY 1980
C
      DIMENSION IPARM(5),IFILE(3),IDCB(144),IRBUF(7200)
      EQUIVALENCE (IPARM(1),IFILE(1))
      DATA LUTTY/1/,LUPRT/6/
C
C---- GET FILE NAME AND PARAMETERS
      CALL RMPAR(IPARM)
C
C---- OPEN VECTOR FILE
   10 CALL OPEN(IDCB,IER,IFILE,2,0,IPARM(4))
      IF(IER .GE. 0) GO TO 20
      WRITE(LUTTY,1000) IFILE,IER
 1000 FORMAT(" OPENING ERROR: FILE=",3A2," IER=",I5)
      STOP
C
C---- RESET ERROR INDICATOR
   20 IOVF=0
C
C---- RASTERIZE AND PLOT
      CALL VRAS(IOVF,IDCB,IRBUF,7200,LUPRT)
      CALL CLOSE(IDCB)
      IF(IOVF .GE. 0) GO TO 30
      WRITE(LUTTY,1010) IOVF
 1010 FORMAT(" RASTERIZING ERROR: ",I5)
      STOP
   30 IF(IPARM(5) .GE. 0) GO TO 40
C
C---- REPLOT FILE?
      WRITE(LUTTY,1020)
 1020 FORMAT(" REPLOT? (YE OR NO) _")
      READ(LUTTY,1030) IOVF
 1030 FORMAT(A2)
      IF(IOVF .EQ. 2HYE) GO TO 10
C
C---- PURGE FILE
   40 CALL PURGE(IDCB,IER,IFILE)
      STOP
      END


  FTN4 COMPILER: HP92060-16092 REV. 1913 (790206)


  **  NO WARNINGS **  NO ERRORS **   PROGRAM = 07513      COMMON = 00000
 PAGE 0002  FTN.    9:49 AM  MON.,  9  MAR., 1981


0046         END$
```

```
 PAGE 0001
0001                      ASMB,R,L,T,C
VRAS   R 000005
.ENTR  X 000001
READF  X 000002
EXEC   X 000003
ICVOF  R 000000
TDCBN  R 000001
BUFF   R 000002
BFSIZ  R 000003
LU     R 000004
ZOPB   R 000034
VECT   R 000056
VECT1  R 000072
NODEA  R 000113
MEMCK  R 000124
VECT2  R 000134
NEXTV  R 000162
VECT3  R 000166
XY     R 000207
VECT5  R 000213
DX     R 000217
VT56C  R 000227
VECT6  R 000232
DY     R 000234
VECT4  R 000245
VECT7  R 000256
VECT8  R 000263
CHAR   R 000266
SM90   R 000326
LARGE  R 000331
TAB0   R 000342
LG90   R 000355
TAB90  R 000357
FILL   R 000366
CCNT   R 000377
DSHFT  R 000421
HDLP   R 000422
LSTB   R 000433
BCK    R 000435
EVWD   R 000441
WHDLF  R 000455
TOBI   R 000462
ROTLB  R 000506
ROTRB  R 000510
EX1    R 000511
EVEN   R 000514
POLL   R 000517
POLL1  R 000523
NEXTR  R 000530
TCHAR  R 000534
CGTO   R 000543
Q1     R 000553
Q2     R 000571
Q3     R 000613
Q4     R 000617
Q25    R 000645
```

```
 PAGE 0002
Q47    R 000652
EX2    R 000664
Q5     R 000667
Q6     R 000710
Q7     R 000714
CHIN   R 000742
CHIN3  R 000755
CHIN4  R 000757
DOBC   R 000771
CHIN5  R 000776
ROTL   R 001021
ROTR   R 001023
CHIN1  R 001026
NOOFF  R 001043
CHIN6  P 001047
CHIN8  R 001057
CHIN7  R 001064
OUT    P 001067
BUFFR  R 001100
FIN    R 001103
INPUT  R 001113
OV     R 001147
OV1    R 001150
IBUFA  R 001152
TBUF   P 001153
TVFFL  R 001557
BRT    R 001560
RASBF  R 001561
IXCON  R 001562
TER    R 001563
LEN    R 001564
X1     R 001565
X2     R 001566
Y1     R 001567
Y2     R 001570
TDX    P 001571
TDY    P 001572
TYDIF  R 001573
LCNT   R 001574
CONWD  R 001575
BTCNT  R 001576
SAVEA  R 001577
BFLWA  R 001600
RQST   P 001601
BADDP  R 001602
LENGT  R 001603
BUF1   P 001604
D2     R 002010
D3     R 002011
RASAD  R 002012
POLPT  R 002013
TBFPT  R 002014
TCVFL  R 002015
RELAD  R 002016
TBL90  R 002017
TBL0   R 002020
```

```
 PAGE 0003
LSLWD  R 002021
COUNT  R 002022
TEMP   R 002023
COUT   R 002024
ROTLW  R 002025
ROTRW  R 002026
CNWD2  R 002027
TBL    R 002030
FSTWD  R 002023
TRFL   R 002031
T0     R 002032
T90    R 002372
**  NO ERRORS PASS#1 **RTE ASMB 760924**
```

```
 PAGE 0004 #01  ** VECTOR TO RASTER PROCESSOR
0001                ASMB,R,L,T,C
0003  00000               NAM VRAS,7
0004*
0005*        VERSION: B        DATE: 750919-2        WORK FILE: VRSRS
0006*
0007*      ERROR RETURNS:
0008*         ICVOF = -1        VECTOR BUFFER OVERFLOW.
0009*
0010*               = -3        FMGR DETECTED AN END-OF-FILE.
0011*
0012*               = ANY OTHER NEGATIVE NUMBER MEANS FMGR ERROR CODE.
0013*
0014*
0015*
0016                      ENT VRAS
0017                      EXT .ENTR,READF,EXEC
0018  00000 000000  ICVOF NOP       OVERFLOW INDICATOR
0019  00001 000000  IDCBN NOP       DATA CONTROL BLOCK.
0020  00002 000000  BUFF  NOP       VECTOR BUFFER.
0021  00003 000000  BFSIZ NOP       VECTOR BUFFER SIZE.
0022  00004 000000  LU    NOP       LU # OF STATUS.
0023  00005 000000  VRAS  NOP
0024  00006 016001X      JSB .ENTR
0025  00007 000000R       DEF ICVOF
0026  00010 062002R       LDA BUFF
0027  00011 073561R       STA RASBF      1ST WORD RASTER BUFFER
0028  00012 142003R       ADA BFSIZ,I    ADD LENGTH.
0029  00013 042632R       ADA =D-1
0030  00014 073600R       STA BFLWA      POINT TO LAST WORD OF BUFFER.
0031  00015 066632R       LDB =B177777        MARK TOP AS BOTTOM
0032  00016 177561R       STB RASBF,I
0033  00017 076015R       STB ICVFL      SET RASTER PROCESS AS ACTIVE
0034  00020 002400        CLA
0035  00021 073601R       STA RQST
0036  00022 073557R       STA IVEFL          INPUT BUFFER EOF FLAG
0037  00023 002404        CLA,INA       SET RELATIVE RECORD # TO
0038  00024 072031R       STA IREL         FIRST RECORD.
0039  00025 016003X      JSB EXEC       INITIALIZE STATUS
0040  00026 000031R       DEF *+3
0041  00027 002011R       DEF D3
0042  00030 100004R       DEF LU,I
0043  00031 017113R      JSB INPUT      DO INITIAL I/O
0044  00032 162014R       LDA IBFPT,I         SET MAX X-VALVE
0045  00033 073562R       STA IXCUR
0046* BEGIN PROCESSING
0047  00034 067602R ZORB  LDB BADDR      CLEAR THE OUTPUT RASTER BUFFER
0048  00035 062633R       LDA =D-132
0049  00036 072022R       STA  COUNT       132 WORDS
0050  00037 002400        CLA
0051  00040 170001        STA 1,I
0052  00041 006004        INB
0053  00042 036022R       ISZ COUNT
0054  00043 026040R      JMP *-3
0055  00044 063562R       LDA IXCUR      SET RASTER POSITION POINTER TO
0056  00045 042632R       ADA =D-1        THE NEXT LINE
0057  00046 073562R       STA IXCUR
```

```
00047 063557R       LDA TVFFL      SORTED VECTOR FILE ALL READ IN
00050 002021        SSA, RSS
00051 026056R       JMP VECT       IF NOT, TEST THEM AGAINST TXCUR
00052 062015R       LDA TCVFL      RASTER PROCESS FILE EMPTY
00053 002002        SZA
00054 026517R       JMP POLL       IF NOT, PROCESS THE NEXT RASTER
00055 027103R       JMP FIN        CLOSEOUT PLOTTER
00056 162014R VECT  LDA TBFPT, I         END OF INPUT DATA?
00057 022634R       XOR =B77776
00060 002002        SZA
00061 026064R       JMP *+3
00062 017113R       JSB INPUT     INPUT NEW DATA, IF FOR ENCOUNTER
00063 026056R       JMP VECT
00064 042632R       ADA =D-1    TEST FOR EOF IN INPUT DATA
00065 002002        SZA
00066 026072R       JMP VECT1
00067 003000        CMA
00070 073557R       STA TVFFL      SET TVFFL FOR VECTOR FILE EMPTY
00071 026517R       JMP POLL
00072 162014R VECT1 LDA TBFPT,I     COMPARE CURRENT VECTOR AGAINST
00073 003004        CMA, INA        CURRENT LINE POSITION
00074 043562R       ADA TXCUR
00075 002021        SSA, RSS
00076 026517R       JMP POLL
00077 062014R       LDA TBFPT      ENTER A NEW VECTOR INTO THE
00100 073565R       STA X1         RASTER BUFFER
00101 002004        INA      SETUP WORKING VALUES
00102 073567R       STA Y1
00103 002004        INA
00104 073566R       STA X2
00105 002004        INA
00106 073570R       STA Y2
00107 163566R       LDA X2,I       CHECK FOR HORIZONTAL LINE
00110 153565R       CPA X1,I       IF X1=X2
00111 026366R       JMP FILL       ENTER THE LINE
00112 067561R       LDB RASBF      OTHERWISE, FIND A 6 WORD SLOT IN
00113 076012R NODEA STB RASAD     THE PROCESS BUFFER
00114 160001        LDA 1,I
00115 002003        SZA,RSS        IF ZERO, THIS IS A RELEASED
00116 026134R       JMP VECT2      BLOCK SO USE IT
00117 022632R       XOR =B177777      TEST FOR BOTTOM OF BUFFER
00120 002003        SZA,RSS         IF IT IS TEST FOR LWAM
00121 026124R       JMP MEMCK
00122 046635R       ADB =D6        INCREMENT THE RASTER BUFFER
00123 026113R       JMP NODEA       CONTINUE SEARCH
00124 046635R MEMCK ADB =D6
00125 074000        STB 0          TEST FOR END OF BUFFER AREA
00126 007004        CMB,INB
00127 047600R       ADB BFLWA
00130 006020        SSB
00131 027147R       JMP OV          OVERFLOW, EXIT
00132 066632R       LDB =B177777      MARK NEW RASTER BOTTOM
00133 174000        STB 0,I
00134 163566R VECT2 LDA X2,I       TEST FOR CHARACTER
00135 052636R       CPA =B77774
00136 026266R       JMP CHAR
```

```
00137 003004        CMA,INA
00140 143565R       ADA X1,I      X1-X2
00141 060012R       LDB RASAD
00142 046637R       ADB =D5
00143 170001        STA 1,I       WORD 6=X1-X2
00144 073571R       STA IDX        ALSO IDX
00145 046640R       ADB =D-4      WORD 2=Y1
00146 163567R       LDA Y1,I
00147 170001        STA 1,I
00150 003004        CMA,INA       IYDIF=Y2-Y1
00151 143570R       ADA Y2,I
00152 073573R       STA IYDIF
00153 002002        SZA           TEST FOR VERTICAL LINE
00154 026166R       JMP VECT3
00155 002004        INA
00156 172012R       STA RASAD,I
00157 006004        INB
00160 063571R       LDA IDX       WORD 3=X1-X2  (COUNTER)
00161 170001        STA 1,I
00162 062014R NEXTV LDA IBFPT     INCREMENT INPUT BUFFER TO NEXT
00163 042641R       ADA =D4       VECTOR
00164 072014R       STA IBFPT
00165 026056R       JMP VECT      AND SEE IF IT IS READY
00166 002021  VECT3 SSA,RSS
00167 026171R       JMP *+2       TAKE ABS(Y2-Y1)
00170 003004        CMA,INA       TAKE COMPLEMENT
00171 073572R       STA IDY
00172 046642R       ADB =D3
00173 170001        STA 1,I        WORD 5=IDY
00174 063573R       LDA IYDIF
00175 002020        SSA
00176 026245R       JMP VECT4
00177 063571R       LDA IDX       QUADRANT IDENTIFICATION
00200 003004        CMA,INA
00201 043572R       ADA IDY         IDY-IDX
00202 060012R       LDB RASAD     WORD 1 ADDRESS TO CONTAIN
00203 002002        SZA      QUADRANT
00204 026213R       JMP VECT5
00205 062635R       LDA =D6       WORD 1=6  QUADRANT 6
00206 170001        STA 1,I
00207 046643R XY    ADB =D2
00210 063572R       LDA IDY       WORD 3=IDY
00211 170001        STA 1,I
00212 026162R       JMP NEXTV
00213 002020  VECT5 SSA
00214 026232R       JMP VECT6     QUAD 5
00215 062644R       LDA =D7       QUAD 7
00216 170001        STA 1,I
00217 046643R DX    ADB =D2       WORD 3=IDY
00220 063572R       LDA IDY
00221 170001        STA 1,I
00222 003004        CMA,INA       WORD 4=2*IDX-IDY
00223 073572R       STA IDY
00224 063571R       LDA IDX
00225 001000        ALS      2*IDX
00226 043572R       ADA IDY   -IDY
```

```
0170 00227 006004  VT56C INB      (B)=A(WORD#4)
0171 00230 170001        STA 1,I
0172 00231 026162R       JMP NEXTV
0173 00232 062637R VECT6 LDA =D5      QUADRANT 5
0174 00233 170001        STA 1,I      WORD 1=5
0175 00234 046643R DY    ADB =D2      WORD 3=IDX
0176 00235 063571R       LDA IDX
0177 00236 170001        STA 1,I
0178 00237 003004        CMA,INA      WORD 4=2*IDY-IDX
0179 00240 073571R       STA IDX       -IDX
0180 00241 063572R       LDA IDY
0181 00242 001000        ALS     2*IDY
0182 00243 043571R       ADA IDX
0183 00244 026227R       JMP VT56C
0184 00245 066012R VECT4 LDB RASAD    (B)=A(WORD#1)
0185 00246 063571R       LDA IDX
0186 00247 003004        CMA,INA
0187 00250 043572R       ADA IDY       IDY-IDX
0188 00251 002002        SZA     IDY-IDX=0?
0189 00252 026256R       JMP VECT7
0190 00253 062642R       LDA =D3      YES, QUADRANT 3
0191 00254 170001        STA 1,I
0192 00255 026207R       JMP XY
0193 00256 002020  VECT7 SSA     IDY-IDX>0 ?
0194 00257 026263R       JMP VECT8    NO, QUADRANT 2
0195 00260 062641R       LDA =D4      YES, QUADRANT 4
0196 00261 170001        STA 1,I
0197 00262 026217R       JMP DX
0198 00263 062643R VECT8 LDA =D2      QUADRANT 2
0199 00264 170001        STA 1,I
0200 00265 026234R       JMP DY
0201* PROCESS A CHARACTER ENTRY INTO THE RASTER PROCESS BUFFER
0202 00266 066012R CHAR  LDB RASAD
0203 00267 062645R       LDA =D8
0204 00270 170001        STA 1,I      WORD 1=8  CHARACTER
0205 00271 006004        INB
0206 00272 163567R       LDA Y1,I
0207 00273 170001        STA 1,I      WORD 2= Y
0208 00274 163570R       LDA Y2,I
0209 00275 012646R       AND =B1      LEFT OR RIGHT BYTE OF CHARACTER
0210 00276 046642R       ADB =D3       0=LEFT
0211 00277 170001        STA 1,I       1=RIGHT
0212 00300 163570R       LDA Y2,I
0213 00301 012647R       AND =B76     GET RELATIVE ADDRESS INDEX
0214 00302 001100        ARS
0215 00303 072016R       STA RELAD    SAVE FOR LATER USE
0216 00304 163570R       LDA Y2,I       GET SIZE AND ORIENTATION
0217 00305 012650R       AND =B1400
0218 00306 001700        ALF
0219 00307 001222        RAL,RAL
0220 00310 072023R       STA TEMP
0221 00311 006004        INB
0222 00312 002020        SSA     LARGE OR SMALL ?
0223 00313 026331R       JMP LARGE
0224 00314 002400        CLA     WORD 6=0 SMALL
0225 00315 170001        STA 1,I
```

```
 PAGE 0008 #01  ** VECTOR TO RASTER PROCESSOR
0226  00316 046651R       ADB =D-3      SET UP ORIENTATION AND COUNT
0227  00317 062023R       LDA TEMP
0228  00320 001200        RAL
0229  00321 002020        SSA
0230  00322 026326R       JMP SM90
0231  00323 062644R       LDA =D7       0 DEGREE ORIENTATION
0232  00324 170001        STA 1,I       WORD 3=7 COUNT WORD
0233  00325 026342R       JMP TAB0       DETERMINE CHARACTER ADDRESS
0234  00326 062652R SM90  LDA =D-5      90 DEGREE ORIENTATION
0235  00327 170001        STA 1,I       WORD 3=-5 COUNT WORD
0236  00330 026357R       JMP TAB90      DETERMINE CHARACTER ADDRESS
0237  00331 002404  LARGE CLA,INA
0238  00332 170001        STA 1,I       WORD 6=1 LARGE
0239  00333 046651R       ADB =D-3       ORIENTATION
0240  00334 062023R       LDA TEMP
0241  00335 001200        RAL
0242  00336 002020        SSA
0243  00337 026355R       JMP LG90
0244  00340 062653R       LDA =D14      0 DEGREES, COUNT=14
0245  00341 170001        STA 1,I
0246  00342 006004  TAB0  INB       CHARACTER WORD ADDRESS IN
0247  00343 062016R       LDA RELAD     WORD 4
0248  00344 001000        ALS      7*RELAD+A(0 DEG CHAR TAB)
0249  00345 042016R       ADA RELAD
0250  00346 072023R       STA TEMP
0251  00347 062016R       LDA RELAD
0252  00350 001020        ALS,ALS
0253  00351 042023R       ADA TEMP
0254  00352 042020R       ADA TBL0
0255  00353 170001        STA 1,I
0256  00354 026162R       JMP NEXTV
0257  00355 062654R LG90  LDA =D-10      SET LARGE COUNT 90 DEGREES
0258  00356 170001        STA 1,I
0259  00357 006004  TAB90 INB       CHARACTER WORD ADDRESS
0260  00360 062016R       LDA RELAD      5*RELAD+A(90 DEGREE CHARACTER
0261  00361 001020        ALS,ALS         TABLE)
0262  00362 042016R       ADA RELAD
0263  00363 042017R       ADA TBL90
0264  00364 170001        STA 1,I
0265  00365 026162R       JMP NEXTV
0266* PLACE A HORIZONTAL LINE  IN THE OUTPUT BUFFER FOR THE
0267  00366 163567R FILL  LDA Y1,I        CURRENT RASTER
0268  00367 167570R       LDB Y2,I
0269  00370 007004        CMB,INB
0270  00371 040001        ADA 1
0271  00372 002020        SSA      Y2>Y1?
0272  00373 026377R       JMP CCNT    YES
0273  00374 167570R       LDB Y2,I    NO,EXCHANGE
0274  00375 177567R       STB Y1,I
0275  00376 003004        CMA,INA
0276  00377 042632R CCNT   ADA =D-1      INSURE 1 POINT IS PLOTTED
0277  00400 072022R       STA COUNT     COMPUTE FIRST BUFFER WORD WHERE
0278  00401 163567R       LDA Y1,I        A POINT NEED TO BE
0279  00402 006400        CLB      SAVE LOW ORDER 4 BITS
0280  00403 101104        RRR 4
0281  00404 043602R       ADA BADDR      ADD FIRST WORD OF BUFFER FOR
```

```
00405 072023R         STA FSTWD      INITIAL WORD
00406 002400          CLA       GET BIT POSITION OF 1ST OFFSET
00407 100104          RRL 4
00410 032021R         IOR LSLWD      TO THAT BIT
00411 072421R         STA OSHFT      SAVE SHIFT CODE
00412 012655R         AND =B17
00413 000000          NOP
00414 002003          SZA,RSS
00415 026441R         JMP EVWD
00416 042656R         ADA =D-16
00417 073576R         STA BTCNT      SAVE FIRST WORD WORD BIT COUNT
00420 162023R         LDA FSTWD,I
00421 000000  OSHFT   NOP       SHIFT OFFSET
00422 032657R HDLP    IOR =B100000        SET THE BIT
00423 100041          LSL 1
00424 036022R         ISZ COUNT      INCREMENT TOTAL COUNT
00425 026435R         JMP BCK        IF LAST BIT, SHIFT OUT UNUSED
00426 037576R         ISZ BTCNT      BITS
00427 026431R         JMP *+2
00430 026433R         JMP LSTB
00431 100041          LSL 1
00432 026426R         JMP *-4
00433 176023R LSTB    STB FSTWD,I       SAVE THE LAST WORD AND RETURN
00434 026162R         JMP NEXTV      FOR NEXT VECTOR IF ANY
00435 037576R BCK     ISZ BTCNT      TEST FOR LAST BIT IN WORD
00436 026422R         JMP HDLP
00437 176023R         STB FSTWD,I         IF SO SAVE WORD
00440 036023R         ISZ FSTWD
00441 062022R EVWD    LDA COUNT      TEST FOR 16 BITS LEFT
00442 002003          SZA,RSS
00443 026162R         JMP NEXTV      OR IF COMPLETED
00444 042660R         ADA =D16
00445 002020          SSA
00446 026455R         JMP WHOLE
00447 002003          SZA,RSS        IF LESS THAN 16 BITS DO HEAD
00450 026455R         JMP WHOLE      LOOPS
00451 162023R         LDA FSTWD,I         SET UP FOR LAST WORD
00452 066656R         LDB =D-16
00453 077576R         STB BTCNT
00454 026422R         JMP HDLP
00455 072022R WHOLE   STA COUNT
00456 062632R         LDA =B177777        PLACE ALL BITS ON
00457 172023R         STA FSTWD,I
00460 036023R         ISZ FSTWD
00461 026441R         JMP EVWD
00462 000000  TORI   NOP       PLACE Y IN THE OUTPUT BUFFER
00463 073577R       STA SAVEA     SAVE A
00464 062013R       LDA POLPT
00465 002004        INA
00466 164000        LDB 0,I        GET Y
00467 005121        BRS,BRS
00470 005121        BRS,BRS
00471 047602R       ADB BADDR         WORD ADDRESS
00472 076023R       STB TEMP
00473 160000        LDA 0,I
00474 012655R       AND =B17
```

```
 PAGE 0010 #01  ** VECTOR TO RASTER PROCESSOR
0338   00475 002003          SZA,RSS    TEST FOR NO OFFSET
0339   00476 026514R         JMP EVEN
0340   00477 070001          STA 1      BIT POSITION IN WORD
0341   00500 032025R         IOR ROTLW        OFFSET TO SIGN
0342   00501 072506R         STA ROTLB
0343   00502 074000          STB 0
0344   00503 032026R         IOR ROTRW        RESTORE SHIFT
0345   00504 072510R         STA ROTRB
0346   00505 162023R         LDA TEMP,I
0347   00506 000000  ROTLB NOP       SHIFT TO SIGN
0348   00507 032657R          IOR =B100000    ENTER BIT
0349   00510 000000  ROTRB NOP        RESTORE
0350   00511 172023R EX1   STA TEMP,I
0351   00512 063577R         LDA SAVEA
0352   00513 126462R         JMP TOBI,I
0353   00514 162023R EVEN  LDA TEMP,I
0354   00515 032657R          IOR =B100000
0355   00516 026511R          JMP EX1
0356*
0357* POLL THE RASTER BUFFER FOR BITS
0358*
0359   00517 002400  POLL  CLA        SET RASTER BUFFER INACTIVE FLAG
0360   00520 072015R         STA ICVEL        AS INACTIVE
0361   00521 063561R         LDA RASBF
0362   00522 072013R         STA POLPT      INITIALIZE POLL POINTER TO TOP OF
0363   00523 162013R POLL1 LDA POLPT,I          BUFFER
0364   00524 052632R         CPA =B177777         BOTTOM OF PROCESS BUFFER?
0365   00525 027067R         JMP OUT
0366   00526 002002          SZA        TEST FOR COMPLETED SIX WORD BLOCK
0367   00527 026534R         JMP TCHAR
0368   00530 066013R NEXTP LDB POLPT       IF SO, CHECK THE NEXT BLOCK
0369   00531 046635R         ADB =D6
0370   00532 076013R         STB POLPT
0371   00533 026523R         JMP POLL1
0372   00534 052645R TCHAR CPA =D8        CHARACTER ?
0373   00535 026742R         JMP CHIN            YES
0374   00536 016462R         JSB TOBI       PLACE THE FIRST BIT IN
0375   00537 062013R         LDA POLPT        THE OUTPUT BUFFER
0376   00540 002004          INA           (A)=A(Y VALUE)
0377   00541 166013R         LDB POLPT,I
0378   00542 047560R         ADB BRT          GO TO THE APPROPRIATE
0379   00543 124001  CGTO  JMP 1,I           QUADRANT
0380   00544 026553R         JMP Q1
0381   00545 026571R         JMP Q2
0382   00546 026613R         JMP Q3
0383   00547 026617R         JMP Q4
0384   00550 026667R         JMP Q5
0385   00551 026710R         JMP Q6
0386   00552 026714R         JMP Q7
0387   00553 066632R Q1    LDB =D-1       SET PROCESS BUFFER
0388   00554 076015R         STB ICVEL      FLAG AS ACTIVE
0389   00555 066013R         LDB POLPT
0390   00556 046643R         ADB =D2        (POLPT)+2
0391   00557 160001          LDA 1,I
0392   00560 042632R         ADA =D-1       DECREMENT COUNT
0393   00561 170001          STA 1,I
```

```
0394  00562 002003          SZA,RSS
0395  00563 026567R         JMP *+4
0396  00564 002021          SSA,RSS
0397  00565 026530R         JMP NEXTP      IF NOT DONE, GO
0398  00566 002400          CLA        ELSE MARK AS COMPLETE
0399  00567 172013R         STA POLPT,I
0400  00570 026530R         JMP NEXTP       DO NEXT ENTRY IN PROCESS BUFFER
0401  00571 042643R Q2      ADA =D2        TEST THE DEVIATION FUNCTION
0402  00572 164000          LDB 0,I         (POLPT)+3
0403  00573 006020          SSB
0404  00574 026645R         JMP Q25
0405  00575 076023R         STB TEMP       RECOMPUTE DEVIATION
0406  00576 002004          INA       (A+3)+2*(A+4)-2*(A+5)
0407  00577 164000          LDB 0,I
0408  00600 005000          BLS
0409  00601 046023R         ADB TEMP
0410  00602 076023R         STB TEMP
0411  00603 002004          INA
0412  00604 164000          LDB 0,I
0413  00605 005000          BLS
0414  00606 007004          CMB,INB
0415  00607 046023R         ADB TEMP
0416  00610 042661R         ADA =D-2       (A)=(POLPT)+3
0417  00611 174000          STB 0,I
0418  00612 042661R         ADA =D-2       Y VALUE ADDRESS (POLPT)+1
0419  00613 164000  Q3      LDB 0,I          DECREMENT Y
0420  00614 046632R         ADB =D-1
0421  00615 174000          STB 0,I
0422  00616 026553R         JMP Q1
0423  00617 164000  Q4      LDB 0,I        (A)=(POLPT)+1
0424  00620 046632R         ADB =D-1        DECREMENT Y VALUE
0425  00621 174000          STB 0,I
0426  00622 042643R         ADA =D2         (POLPT)+3
0427  00623 164000          LDB 0,I         TEST THE DEVIATION FUNCTION
0428  00624 006021          SSB,RSS
0429  00625 026652R         JMP Q47
0430  00626 016462R         JSB TURI       IF LESS THAN ZERO COMPUTE NEW
0431  00627 042643R         ADA =D2       DEVIATION AFTER PLACING BIT
0432  00630 164000          LDB 0,I
0433  00631 005000          BLS
0434  00632 042661R         ADA =D-2
0435  00633 144000          ADB 0,I
0436  00634 174000          STB 0,I       ((POLPT)+3=((POLPT)+3)+
0437  00635 042632R         ADA =D-1        2*((POLPT)+5)
0438  00636 164000          LDB 0,I
0439  00637 046632R         ADB =D-1       DECREMENT COUNT
0440  00640 006003          SZB,RSS         IF NOT ZERO CHECK DEVIATION
0441  00641 026553R         JMP Q1         AGAIN
0442  00642 174000          STB 0,I        SET WORD POINTER TO Y VALUE
0443  00643 042632R         ADA =D-1
0444  00644 026617R         JMP Q4
0445  00645 002004  Q25     INA       COMPUTE NEW DEVIATION
0446  00646 164000          LDB 0,I
0447  00647 005000          BLS
0448  00650 042632R         ADA =D-1
0449  00651 026664R         JMP EX2
```

```
00652 002004  Q47    INA        COMPUTE NEW DEVIATION
00653 164000         LDB 0,I
00654 005000         BLS
00655 007004         CMB,INB   ((POLPT)+3)=((POLPT)+3)+2((POLPT)+5)
00656 076023R        STB TEMP
00657 002004         INA        -2*((POLPT)+4)
00660 164000         LDB 0,I
00661 005000         BLS
00662 046023R        ADB TEMP
00663 042661R        ADA =D-2
00664 144000  EX2    ADB 0,I
00665 174000         STB 0,I
00666 026553R        JMP Q1
00667 042643R Q5     ADA =D2
00670 164000         LDB 0,I      ((POLPT)+3)=((POLPT)+3)+2((POLPT)+4)
00671 006020         SSB
00672 026645R        JMP Q25          -2((POLPT)+5)
00673 002004         INA
00674 164000         LDB 0,I
00675 005000         BLS
00676 076023R        STB TEMP
00677 002004         INA
00700 164000         LDB 0,I
00701 005000         BLS
00702 007004         CMB,INB
00703 046023R        ADB TEMP
00704 042661R        ADA =D-2
00705 144000         ADB 0,I
00706 174000         STB 0,I
00707 042661R        ADA =D-2        POLPT+1
00710 164000  Q6     LDB 0,I
00711 006004         INB        INCREMENT COUNT
00712 174000         STB 0,I
00713 026553R        JMP Q1
00714 164000  Q7     LDB 0,I
00715 006004         INB
00716 174000         STB 0,I
00717 042643R        ADA =D2         (POLPT)+3
00720 164000         LDB 0,I
00721 006021         SSB,RSS
00722 026652R        JMP Q47
00723 016462R        JSB TOBI
00724 042643R        ADA =D2        (POLPT)+5
00725 164000         LDB 0,I
00726 005000         BLS
00727 042661R        ADA =D-2
00730 144000         ADB 0,I
00731 174000         STB 0,I      ((POLPT)+3)=((POLPT)+3)+2*((POLPT)+5)
00732 042632R        ADA =D-1
00733 164000         LDB 0,I        ((POLPT)+2)
00734 046632R        ADB =D-1
00735 006003         SZB,RSS
00736 026553R        JMP Q1
00737 174000         STB 0,I
00740 042632R        ADA =D-1          (POLPT)+1
00741 026714R        JMP Q7
```

```
*  CHARACTER PROCESSOR ROUTINE
   00742 062013R CHIN  LDA POLPT    GET CHARACTER ADDRESS
   00743 042642R       ADA =D3
   00744 164000        LDB 0,I
   00745 076024R       STB COUT       SAVE IT
   00746 002004        INA
   00747 164000        LDB 0,I       LEFT OR RIGHT BYTE
   00750 162024R       LDA COUT,I
   00751 006002        SZB
   00752 026755R       JMP CHIN3
   00753 012662R       AND =B177400       MASK OFF RIGHT BYTE
   00754 026757R       JMP CHIN4
   00755 012663R CHIN3 AND =B377     MASK OFF LEFT BYTE
   00756 001727        ALF, ALF       AND POSITION AT SIGN
   00757 072024R CHIN4 STA COUT
   00760 066013R       LDB POLPT
   00761 046637R       ADB =D5
   00762 160001        LDA 1,I       TEST FOR LARGE OR SMALL
   00763 002003        SZA,RSS
   00764 026776R       JMP CHIN5     IF LARGE DOUBLE CHARACTER
   00765 066664R       LDB =D-8
   00766 077574R       STB LCNT       SET DOUBLER COUNT
   00767 066024R       LDB COUT
   00770 005727        BLF,BLF
   00771 101041 DUBC   LSR 1         DOUBLE CHARACTER
   00772 001100        ARS
   00773 037574R       ISZ LCNT
   00774 026771R       JMP DUBC
   00775 072024R       STA COUT
   00776 062013R CHIN5 LDA POLPT
   00777 002004        INA
   01000 164000        LDB 0,I     DETERMINE LEFT MOST BIT POSITION
   01001 005121        BRS,BRS       DETERMINE RELATIVE WORD
   01002 005121        BRS,BRS
   01003 047602R       ADB BADDR      SAVE THE LEFT WORD ADDRESS
   01004 076023R       STB TEMP
   01005 160000        LDA 0,I
   01006 012655R       AND =B17       COMPUTE OFFSET
   01007 002003        SZA,RSS     IF NO OFFSET DO SPECIAL
   01010 027043R       JMP NOOFF
   01011 070001        STA 1         SAVE AT B
   01012 032025R       IOR ROTLW       LEFT SHIFT TO SIGN INSTRUCTION
   01013 073021R       STA ROTL
   01014 074000        STB 0
   01015 032026R       IOR ROTRW   RIGHT SHIFT RESTORE INSTRUCTION
   01016 073023R       STA ROTR
   01017 104200        DLD TEMP,I
   01020 102023R
   01021 000000 ROTL   NOP       OFFSET
   01022 032024R       IOR COUT
   01023 000000 ROTR   NOP       OFFSET
   01024 104400        DST TEMP,I
   01025 102023R
   01026 062013R CHIN1 LDA POLPT     ADJUST COUNT
   01027 042643R       ADA =D2
   01030 164000        LDB 0,I
```

```
 PAGE 0014 #01  ** VECTOR TO RASTER PROCESSOR
0560   01031 006020         SSB
0561   01032 027035R        JMP *+3
0562   01033 046632R        ADB =D-1
0563   01034 027036R        JMP *+2
0564   01035 006004         INB
0565   01036 174000         STB 0,I      SAVE NEW COUNT
0566   01037 006002         SZB
0567   01040 027047R        JMP CHTN6
0568   01041 176013R         STB POLPT,I      MARK AS COMPLETED
0569   01042 026530R        JMP NEXTR
0570   01043 162023R NOOFF  LDA TEMP,I
0571   01044 032024R        IOR COUNT
0572   01045 172023R        STA TEMP,I
0573   01046 027026R        JMP CHTN1
0574   01047 042642R CHTN6  ADA =D3
0575   01050 164000         LDB 0,I         LARGE OR SMALL
0576   01051 006003         SZB,RSS
0577   01052 027057R        JMP CHTN8
0578   01053 042651R        ADA =D-3         LARGE, INCREMENT CHARACTER
0579   01054 164000         LDB 0,I      ONLY ON EVEN COUNTS
0580   01055 004010         SLB
0581   01056 027064R        JMP CHTN7
0582   01057 062013R CHTN8  LDA POLPT    INCREMENT CHARACTER ADDRESS
0583   01060 042642R        ADA =D3
0584   01061 164000         LDB 0,I
0585   01062 006004         INB
0586   01063 174000         STB 0,I
0587   01064 066632R CHTN7  LDB =D-1         SET PROCESS BUFFER AS ACTIVE
0588   01065 076015R        STB ICVFL
0589   01066 026530R        JMP NEXTR
0590*  SEND DATA TO THE STATOS
0591   01067 062665R OUT    LDA =B100      SET BINARY(PLOT) BIT.
0592   01070 132004R        IOR LU,I       MERGE LU #.
0593   01071 073575R        STA CONWD      SET IN CONTROL WORD.
0594   01072 063602R        LDA BADDR      GET BUFFER ADDRESS.
0595   01073 073100R        STA BUFFR      SET IN EXEC  CALL.
0596*
0597   01074 016003X        JSB EXEC       EXEC CALL
0598   01075 001102R        DEF *+5          TO PLOT
0599   01076 002010R        DEF D2           ON STATOS.
0600   01077 001575R        DEF CONWD
0601   01100 001100R BUFFR  DEF *
0602   01101 001603R        DEF LENGT
0603   01102 026034R        JMP ZUPB
0604*  CLOSE OUT THE PLOTTER
0605   01103 062666R FIN    LDA =B1600     SET SLEW REQUEST.
0606   01104 132004R        IOR LU,I       MERGE LU #.
0607   01105 073575R        STA CONWD      SET CONTROL WORD.
0608*
0609   01106 016003X        JSB EXEC       EXEC CALL TO
0610   01107 001112R        DEF *+3          SLEW ON STATOS.
0611   01110 002011R        DEF D3
0612   01111 001575R        DEF CONWD
0613   01112 126005R        JMP VRAS,I
0614*
0615   01113 000000  INPUT  NOP           INPUT 256 WORDS FROM THE DISC
```

```
0616  01114 063152R        LDA IBUFA
0617  01115 066667R        LDB =D-260        PUT EOR'S IN THE INPUT FILE
0618  01116 076022R        STB COUNT
0619  01117 066634R        LDB =B77776
0620  01120 174000         STB 0,I
0621  01121 002004         INA
0622  01122 036022R        ISZ COUNT
0623  01123 027120R        JMP *-3
0624*
0625  01124 016002X        JSB READF         CALL FMGR TO
0626  01125 001134R        DEF *+7             READ A RECORD.
0627  01126 100001R        DEF IDCBN,I
0628  01127 001563R        DEF IER
0629  01130 001153R        DEF IBUF
0630  01131 002030R        DEF IBL
0631  01132 001564R        DEF LEN
0632  01133 002031R        DEF IRFL
0633*
0634  01134 062651R        LDA =D-3       SET EOF ERROR CODE.
0635  01135 067564R        LDB LEN        CHECK FOR END-OF-FILE.
0636  01136 006020         SSB
0637  01137 027150R          JMP OV1          EOF.
0638*
0639  01140 063563R        LDA IER        SET FMGR ERROR CODE.
0640  01141 002020         SSA            ERROR?
0641  01142 027150R          JMP OV1        YES.
0642*
0643  01143 036031R        ISZ IRFL       BUMP RECORD #.
0644  01144 063152R        LDA IBUFA      INITIALIZE INPUT BUFFER POINTER
0645  01145 072014R        STA IBFPT
0646  01146 127113R        JMP INPUT,I
0647  01147 062632R OV     LDA =D-1
0648  01150 172000R OV1    STA ICVOF,I      SAVE ERROR CODE.
0649  01151 126005R        JMP VRAS,I
0650*
0651  01152 001153R IBUFA DEF *+1
0652  01153 000000  IBUF   BSS 260        INPUT BUFFER
0653  01557 000000  IVFFL  BSS 1          IBUF EOF FLAG
0654  01560 000543R BRT    DEF CGTO       COMPUTED GO TO BRANCH ADDRESS
0655  01561 000000  RASBF  BSS 1          RASTER BUFFER FIRST WORD
0656  01562 000000  IXCOO  BSS 1          FIRST RASTER POSITION
0657  01563 000000  IER    BSS 1          ERROR CODE FROM FMGR.
0658  01564 000000  LEN    BSS 1          EOF INDICATOR FROM FMGR.
0659  01565 000000  X1     BSS 1
0660  01566 000000  X2     BSS 1
0661  01567 000000  Y1     BSS 1
0662  01570 000000  Y2     BSS 1
0663  01571 000000  IDX    BSS 1
0664  01572 000000  IDY    BSS 1
0665  01573 000000  IYDIF  BSS 1
0666  01574 000000  LCNT   BSS 1
0667  01575 000000  CONWD  BSS 1            CONTROL WORD.
0668  01576 000000  BTCNT  BSS 1
0669  01577 000000  SAVEA  BSS 1
0670  01600 000000  BFLWA  BSS 1            LAST WORD OF BUFFER AREA.
0671  01601 000000  RQST   NOP
```

```
 PAGE 0016 #01  ** VECTOR TO RASTER PROCESSOR
0672  01602 001604R BADDR DEF BUF1
0673  01603 000204  LENGT DEC 132        BUFFER LENGTH.
0674  01604 000000  BUF1  BSS 132        BUFFER
0675  02010 000002  D2    DEC 2          REQUEST WRITE IN EXEC.
0676  02011 000003  D3    DEC 3          CONTROL REQUEST IN EXEC.
0677  02012 000000  RASAD BSS 1         PROCESS BUFFER ENTRY POINTER
0678  02013 000000  POLPT BSS 1     PULL POINTER IN PROCESS BUFFER
0679  02014 000000  IBFPT BSS 1         INPUT BUFFER POINTER
0680  02015 000000  TCVFL BSS 1         RASTER PROCESS FILE EMPTY FLAG
0681  02016 000000  RELAD BSS 1         CHARACTER RELATIVE ADDRESS
0682  02017 002372R TBL90 DEF T90       ADDRESS OF 90 DEGREE CHAR TABLE
0683  02020 002032R TBL0  DEF T0        ADDRESS OF 0  DEGREE CHAR TABLE
0684  02021 100040  LSLWD OCT 100040
0685  02022 000000  COUNT BSS 1         FILL BITS TOTAL COUNT
0686  02023 000000  TEMP BSS 1
0687  02024 000000  COUT  BSS 1         OUTPUT CHARACTER
0688  02025 100100  ROTLW OCT 100100     CHARACTER ROTATES FOR OFFSET
0689  02026 101100  ROTRW OCT 101100
0690  02027 000102  CNWD2 OCT 102       CONTROL WORD FOR DISC BINARY READ
0691  02030 000400  TBL   DEC 256
0692  02023         FSTWD EQU TEMP
0693  02031 000000  TRFL  NOP
0694  02032 070160  T0    OCT 070160        @A   0 DEGREE TABLE
0695  02033 104210        OCT 104210
0696  02034 004210        OCT 004210
0697  02035 064370        OCT 064370
0698  02036 124210        OCT 124210
0699  02037 124210        OCT 124210
0700  02040 070210        OCT 070210
0701  02041 170160        OCT 170160        BC
0702  02042 104210        OCT 104210
0703  02043 104200        OCT 104200
0704  02044 170200        OCT 170200
0705  02045 104200        OCT 104200
0706  02046 104210        OCT 104210
0707  02047 170160        OCT 170160
0708  02050 170370        OCT 170370        DE
0709  02051 104200        OCT 104200
0710  02052 104200        OCT 104200
0711  02053 104360        OCT 104360
0712  02054 104200        OCT 104200
0713  02055 104200        OCT 104200
0714  02056 170370        OCT 170370
0715  02057 174160        OCT 174160        FG
0716  02060 100210        OCT 100210
0717  02061 100200        OCT 100200
0718  02062 170270        OCT 170270
0719  02063 100210        OCT 100210
0720  02064 100210        OCT 100210
0721  02065 100160        OCT 100160
0722  02066 104370        OCT 104370        HI
0723  02067 104040        OCT 104040
0724  02070 104040        OCT 104040
0725  02071 174040        OCT 174040
0726  02072 104040        OCT 104040
0727  02073 104040        OCT 104040
```

```
0728   02074 104370          OCT 104370
0729   02075 004210          OCT 004210          JK
0730   02076 004220          OCT 004220
0731   02077 004240          OCT 004240
0732   02100 004300          OCT 004300
0733   02101 104240          OCT 104240
0734   02102 104220          OCT 104220
0735   02103 070210          OCT 070210
0736   02104 100210          OCT 100210          LM
0737   02105 100330          OCT 100330
0738   02106 100250          OCT 100250
0739   02107 100210          OCT 100210
0740   02110 100210          OCT 100210
0741   02111 100210          OCT 100210
0742   02112 174210          OCT 174210
0743   02113 104160          OCT 104160          NO
0744   02114 144210          OCT 144210
0745   02115 124210          OCT 124210
0746   02116 114210          OCT 114210
0747   02117 104210          OCT 104210
0748   02120 104210          OCT 104210
0749   02121 104160          OCT 104160
0750   02122 170160          OCT 170160          PQ
0751   02123 104210          OCT 104210
0752   02124 104210          OCT 104210
0753   02125 170210          OCT 170210
0754   02126 100250          OCT 100250
0755   02127 100220          OCT 100220
0756   02130 100150          OCT 100150
0757   02131 170160          OCT 170160          RS
0758   02132 104210          OCT 104210
0759   02133 104200          OCT 104200
0760   02134 170160          OCT 170160
0761   02135 120010          OCT 120010
0762   02136 110210          OCT 110210
0763   02137 104160          OCT 104160
0764   02140 174210          OCT 174210          TU
0765   02141 020210          OCT 020210
0766   02142 020210          OCT 020210
0767   02143 020210          OCT 020210
0768   02144 020210          OCT 020210
0769   02145 020210          OCT 020210
0770   02146 020160          OCT 020160
0771   02147 104210          OCT 104210          VW
0772   02150 104210          OCT 104210
0773   02151 104210          OCT 104210
0774   02152 050250          OCT 050250
0775   02153 050250          OCT 050250
0776   02154 050250          OCT 050250
0777   02155 020210          OCT 020210
0778   02156 104210          OCT 104210          XY
0779   02157 104210          OCT 104210
0780   02160 050120          OCT 050120
0781   02161 020040          OCT 020040
0782   02162 050040          OCT 050040
0783   02163 104040          OCT 104040
```

```
02164 104040          OCT 104040
02165 174000          OCT 174000          Z,SQUARE
02166 004370          OCT 4370
02167 010210          OCT 10210
02170 174210          OCT 174210
02171 040210          OCT 40210
02172 100370          OCT 100370
02173 174000          OCT 174000
02174 000000          OCT 0              DIAMOND,CIRCLE
02175 020160          OCT 20160
02176 070210          OCT 70210
02177 174210          OCT 174210
02200 070210          OCT 70210
02201 020160          OCT 20160
02202 000000          OCT 0
02203 000000          OCT 0             SOLID SQUARE,SOLID CIRCLE
02204 174160          OCT 174160
02205 174370          OCT 174370
02206 174370          OCT 174370
02207 174370          OCT 174370
02210 174160          OCT 174160
02211 000000          OCT 0
02212 000040          OCT 000040        BLANK, !
02213 000040          OCT 000040
02214 000040          OCT 000040
02215 000000          OCT 0
02216 000000          OCT 0
02217 000040          OCT 000040
02220 000040          OCT 000040
02221 050120          OCT 050120          "#
02222 050120          OCT 050120
02223 000330          OCT 000330
02224 000000          OCT 000000
02225 000330          OCT 000330
02226 000120          OCT 000120
02227 000120          OCT 000120
02230 020310          OCT 020310          $%
02231 074310          OCT 074310
02232 120020          OCT 120020
02233 070040          OCT 070040
02234 024100          OCT 024100
02235 170230          OCT 170230
02236 020230          OCT 020230
02237 020140          OCT 020140          &'
02240 050140          OCT 050140
02241 050140          OCT 050140
02242 060000          OCT 060000
02243 124000          OCT 124000
02244 110000          OCT 110000
02245 064000          OCT 064000
02246 010100          OCT 010100          ()
02247 020040          OCT 020040
02250 040020          OCT 040020
02251 040020          OCT 040020
02252 040020          OCT 040020
02253 020040          OCT 020040
```

```
0840   02254 010100        OCT 010100
0841   02255 000000        OCT 000000          *+
0842   02256 124040        OCT 124040
0843   02257 070040        OCT 070040
0844   02260 174370        OCT 174370
0845   02261 070040        OCT 070040
0846   02262 124040        OCT 124040
0847   02263 000000        OCT 000000
0848   02264 000000        OCT 000000          /-
0849   02265 000000        OCT 000000
0850   02266 000000        OCT 000000
0851   02267 000160        OCT 000160
0852   02270 030000        OCT 030000
0853   02271 020000        OCT 020000
0854   02272 040000        OCT 040000
0855   02273 000000        OCT 000000          ./
0856   02274 000010        OCT 000010
0857   02275 000020        OCT 000020
0858   02276 000040        OCT 000040
0859   02277 000100        OCT 000100
0860   02300 000200        OCT 000200
0861   02301 020000        OCT 20000
0862   02302 030040        OCT 030040          01
0863   02303 044140        OCT 044140
0864   02304 044040        OCT 044040
0865   02305 044040        OCT 044040
0866   02306 044040        OCT 044040
0867   02307 044040        OCT 044040
0868   02310 030160        OCT 030160
0869   02311 070160        OCT 70160           23
0870   02312 104210        OCT 104210
0871   02313 004010        OCT 004010
0872   02314 070060        OCT 070060
0873   02315 100010        OCT 100010
0874   02316 100210        OCT 100210
0875   02317 174160        OCT 174160
0876   02320 010370        OCT 10370           45
0877   02321 030200        OCT 30200
0878   02322 050200        OCT 50200
0879   02323 110360        OCT 110360
0880   02324 174010        OCT 174010
0881   02325 010010        OCT 010010
0882   02326 010360        OCT 010360
0883   02327 020370        OCT 020370          67
0884   02330 040010        OCT 040010
0885   02331 100020        OCT 100020
0886   02332 170040        OCT 170040
0887   02333 104100        OCT 104100
0888   02334 104200        OCT 104200
0889   02335 070200        OCT 070200
0890   02336 070160        OCT 070160          89
0891   02337 104210        OCT 104210
0892   02340 104210        OCT 104210
0893   02341 070170        OCT 070170
0894   02342 104010        OCT 104010
0895   02343 104020        OCT 104020
```

```
 PAGE 0020 #01  ** VECTOR TO RASTER PROCESSOR
0896  02344 070040          OCT 070040
0897  02345 000140          OCT 000140         : ;
0898  02346 060140          OCT 060140
0899  02347 060000          OCT 060000
0900  02350 000140          OCT 000140
0901  02351 060140          OCT 060140
0902  02352 060040          OCT 060040
0903  02353 000100          OCT 000100
0904  02354 010000          OCT 010000         < =
0905  02355 020000          OCT 020000
0906  02356 040160          OCT 040160
0907  02357 100000          OCT 100000
0908  02360 040160          OCT 040160
0909  02361 020000          OCT 020000
0910  02362 010000          OCT 010000
0911  02363 040160          OCT 040160         > ?
0912  02364 020210          OCT 020210
0913  02365 010010          OCT 010010
0914  02366 004020          OCT 004020
0915  02367 010040          OCT 010040
0916  02370 020000          OCT 020000
0917  02371 040040          OCT 040040
0918  02372 062374   T90    OCT 62374          @A          90 DEGREE TABLE
0919  02373 111022          OCT 111022
0920  02374 171022          OCT 171022
0921  02375 101022          OCT 101022
0922  02376 076374          OCT 76374
0923  02377 177174          OCT 177174         BC
0924  02400 111202          OCT 111202
0925  02401 111202          OCT 111202
0926  02402 111202          OCT 111202
0927  02403 066104          OCT 66104
0928  02404 177376          OCT 177376         DE
0929  02405 101222          OCT 101222
0930  02406 101222          OCT 101222
0931  02407 101222          OCT 101222
0932  02410 076202          OCT 76202
0933  02411 177174          OCT 177174         FG
0934  02412 011202          OCT 11202
0935  02413 011222          OCT 11222
0936  02414 011222          OCT 11222
0937  02415 001164          OCT 1164
0938  02416 177000          OCT 177000         HI
0939  02417 010202          OCT 10202
0940  02420 010376          OCT 10376
0941  02421 010202          OCT 10202
0942  02422 177000          OCT 177000
0943  02423 060376          OCT 60376          JK
0944  02424 100020          OCT 100020
0945  02425 100050          OCT 100050
0946  02426 100104          OCT 100104
0947  02427 077202          OCT 77202
0948  02430 177376          OCT 177376         LM
0949  02431 100004          OCT 100004
0950  02432 100030          OCT 100030
0951  02433 100004          OCT 100004
```

```
0952  02434 100376        OCT 100376
0953  02435 177174        OCT 177174      NO
0954  02436 002202        OCT 2202
0955  02437 004202        OCT 4202
0956  02440 010202        OCT 10202
0957  02441 177174        OCT 177174
0958  02442 177174        OCT 177174      PQ
0959  02443 011202        OCT 11202
0960  02444 011242        OCT 11242
0961  02445 011302        OCT 11302
0962  02446 006374        OCT 6374
0963  02447 177114        OCT 177114      RS
0964  02450 011222        OCT 11222
0965  02451 031222        OCT 31222
0966  02452 051222        OCT 51222
0967  02453 106144        OCT 106144
0968  02454 001176        OCT 1176        TU
0969  02455 001200        OCT 1200
0970  02456 177200        OCT 177200
0971  02457 001200        OCT 1200
0972  02460 001176        OCT 1176
0973  02461 007376        OCT 7376        VW
0974  02462 030100        OCT 30100
0975  02463 140060        OCT 140060
0976  02464 030100        OCT 30100
0977  02465 007376        OCT 7376
0978  02466 143006        OCT 143006      XY
0979  02467 024010        OCT 24010
0980  02470 010360        OCT 10360
0981  02471 024010        OCT 24010
0982  02472 143006        OCT 143006
0983  02473 151174        OCT 151174      Z, SQUARE
0984  02474 131104        OCT 131104
0985  02475 111104        OCT 111104
0986  02476 115104        OCT 115104
0987  02477 113174        OCT 113174
0988  02500 010070        OCT 10070
0989  02501 034104        OCT 34104
0990  02502 076104        OCT 76104
0991  02503 034104        OCT 34104
0992  02504 010070        OCT 10070
0993  02505 076174        OCT 76174    SOLID SQUARE, SOLID CIRCLE
0994  02506 076174        OCT 76174
0995  02507 076174        OCT 76174
0996  02510 076174        OCT 76174
0997  02511 076070        OCT 76070
0998  02512 000000        OCT 0       BLANK, ^
0999  02513 000000        OCT 0
1000  02514 000276        OCT 276
1001  02515 000000        OCT 0
1002  02516 000000        OCT 0
1003  02517 000050        OCT 50        " #
1004  02520 003356        OCT 3356
1005  02521 000000        OCT 0
1006  02522 003356        OCT 3356
1007  02523 000050        OCT 50
```

```
1008   02524 044306           OCT 44306       $ %
1009   02525 052046           OCT 52046
1010   02526 177020           OCT 177020
1011   02527 052310           OCT 52310
1012   02530 022306           OCT 22306
1013   02531 060000           OCT 60000       & '
1014   02532 116016           OCT 116016
1015   02533 131016           OCT 131016
1016   02534 046000           OCT 46000
1017   02535 120000           OCT 120000
1018   02536 000000           OCT 0           ( )
1019   02537 034202           OCT 34202
1020   02540 042104           OCT 42104
1021   02541 101070           OCT 101070
1022   02542 000000           OCT 0
1023   02543 052020           OCT 52020       * +
1024   02544 034020           OCT 34020
1025   02545 076174           OCT 76174
1026   02546 034020           OCT 34020
1027   02547 052020           OCT 52020
1028   02550 000000           OCT 0           /-
1029   02551 100020           OCT 100020
1030   02552 060020           OCT 60020
1031   02553 020020           OCT 20020
1032   02554 000000           OCT 0
1033   02555 000300           OCT 300         -.
1034   02556 000040           OCT 40
1035   02557 100020           OCT 100020
1036   02560 000010           OCT 10
1037   02561 000006           OCT 6
1038   02562 076000           OCT 76000       01
1039   02563 101204           OCT 101204
1040   02564 101376           OCT 101376
1041   02565 076200           OCT 76200
1042   02566 000000           OCT 0
1043   02567 162104           OCT 162104      23
1044   02570 111202           OCT 111202
1045   02571 111222           OCT 111222
1046   02572 111222           OCT 111222
1047   02573 106154           OCT 106154
1048   02574 030236           OCT 30236       45
1049   02575 024222           OCT 24222
1050   02576 022222           OCT 22222
1051   02577 177222           OCT 177222
1052   02600 020142           OCT 20142
1053   02601 074302           OCT 74302       67
1054   02602 112042           OCT 112042
1055   02603 111022           OCT 111022
1056   02604 111012           OCT 111012
1057   02605 060006           OCT 60006
1058   02606 066014           OCT 66014       89
1059   02607 111022           OCT 111022
1060   02610 111222           OCT 111222
1061   02611 111122           OCT 111122
1062   02612 066074           OCT 66074
1063   02613 000000           OCT 0           : ;
```

```
1064  02614 066266        OCT 66266
1065  02615 066166        OCT 66166
1066  02616 000000        OCT 0
1067  02617 000000        OCT 0
1068  02620 000000        OCT 0          < =
1069  02621 010050        OCT 10050
1070  02622 024050        OCT 24050
1071  02623 042050        OCT 42050
1072  02624 000000        OCT 0
1073  02625 000004        OCT 4          > /
1074  02626 042002        OCT 42002
1075  02627 024262        OCT 24262
1076  02630 010014        OCT 10014
1077  02631 000000        OCT 0
      02632 177777
      02633 177574
      02634 077776
      02635 000006
      02636 077774
      02637 000005
      02640 177774
      02641 000004
      02642 000003
      02643 000002
      02644 000007
      02645 000010
      02646 000001
      02647 000076
      02650 001400
      02651 177775
      02652 177773
      02653 000016
      02654 177766
      02655 000017
      02656 177760
      02657 100000
      02660 000020
      02661 177776
      02662 177400
      02663 000377
      02664 177770
      02665 000100
      02666 001600
      02667 177374
1078                      END VRAS
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

```
.ENTR   00017     00024

=B1     .....     00209

=B100   .....     00591

=B10000 ....      00295     00348     00354     U

=B1400  .....     00217

=B1600  .....     00605

=B17    .....     00287     00337     00543     U

=B17740 ....      00516

=B17777 ....      00031     00098     00109     00323     00364

=B377   .....     00518

=B76    .....     00213

=B77774 ....      00112

=B77776 ....      00066     00619

=D-1    .....     00029     00056     00071     00276     00387     00392
        00420     00424     00437     00439     00443     00448     00498
        00500     00504     00562     00587     00647

=D-10   .....     00257

=D-132  .....     00048

=D-16   .....     00291     00319

=D-2    .....     00416     00418     00434     00459     00476     00479
        00495

=D-260  .....     00617

=D-3    .....     00226     00239     00578     00634

=D-4    .....     00120

=D-5    .....     00234

=D-8    .....     00526

=D14    .....     00244

=D16    .....     00313

=D2     .....     00154     00162     00175     00198     00390     00401
```

```
       00426    00431    00463    00487    00492    00558

=D3    .....    00141    00190    00210    00508    00574    00583

=D4    .....    00134    00195

=D5    .....    00117    00173    00522

=D6    .....    00101    00103    00152    00369

=D7    .....    00160    00231

=D8    .....    00203    00372

BADDR  00672    00047    00281    00334    00540    00594

BCK    00306    00298

BFLWA  00670    00030    00106

BFSIZ  00021    00026

BPT    00654    00378

BTCNT  00668    00292    00299    00306    00320

BUF1   00674    00672

BUFF   00020    00026

BUFFR  00601    00595

CCNT   00276    00272

CGTO   00379    00654

CHAR   00202    00113

CHIN   00507    00373

CHIN1  00557    00573

CHIN3  00518    00515

CHIN4  00520    00517

CHIN5  00535    00525

CHIN6  00574    00567

CHIN7  00587    00581

CHIN8  00582    00577

@CNWD2 00690
```

```
CONWD   00667    00593    00600    00607    00612

COUNT   00685    00049    00053    00277    00297    00310    00322
        00618    00622

COUT    00667    00510    00513    00520    00528    00534    00554
        00571

D2      00675    00599

D3      00676    00041    00611

DOBC    00530    00533

DX      00162    00197

DY      00175    00200

EVEN    00353    00339

EVWD    00310    00290    00326

EX1     00350    00355

EX2     00460    00449

EXEC    00017    00039    00597    00609

FILL    00267    00092

FIN     00605    00064

FSTWD   00692    00282    00293    00304    00308    00309    00318
        00324    00325

HDLP    00295    00307    00321

IBFPT   00679    00044    00065    00077    00082    00133    00135
        00645

IBL     00691    00630

IBUF    00652    00629

IBUFA   00651    00616    00644

ICVFL   00680    00033    00061    00360    00388    00588

ICVOF   00018    00025    00648

IDCRN   00019    00627

IDX     00663    00119    00131    00146    00167    00176    00179
        00182    00185
```

```
IDY      00664    00140    00148    00155    00163    00166    00169
         00180    00187

IER      00657    00626    00639

INPUT    00615    00043    00069    00646

IREL     00693    00038    00632    00643

IVEFL    00653    00036    00058    00075

IXCOD    00656    00045    00055    00057    00079

IYDIF    00665    00125    00143

LARGE    00237    00223

LCNT     00666    00527    00532

LEN      00658    00631    00635

LENGT    00673    00602

LG90     00257    00243

LSLWD    00684    00285

LSTR     00304    00301

LU       00022    00042    00592    00606

MEMCK    00103    00100

NEXTR    00368    00397    00400    00569    00589

NEXTV    00133    00157    00172    00256    00265    00305    00312

NODEA    00094    00102

NOOFF    00570    00545

OSHFT    00294    00286

OUT      00591    00365

OV       00647    00108

OV1      00648    00637    00641

POLL     00359    00063    00076    00081

POLL1    00363    00371

POLPT    00678    00329    00362    00363    00368    00370    00375
```

```
         00377    00389    00399    00507    00521    00535    00557
         00568    00582

Q1       00387    00380    00422    00441    00462    00483    00502

Q2       00401    00381

Q25      00445    00404    00466

Q3       00419    00382

Q4       00423    00383    00444

Q47      00450    00429    00490

Q5       00463    00384

Q6       00480    00385

Q7       00464    00386    00505

RASAD    00677    00094    00116    00129    00149    00184    00202

RASBF    00655    00027    00032    00093    00361

READF    00017    00625

RFLAD    00681    00215    00247    00249    00251    00260    00262

ROTL     00553    00548

ROTLB    00347    00342

ROTLW    00688    00341    00547

ROTR     00555    00551

ROTRB    00349    00345

ROTRW    00689    00344    00550

ROST     00671    00035

SAVEA    00669    00328    00351

SM90     00234    00230

T0       00694    00683

T90      00918    00682

TAB0     00246    00233

TAB90    00259    00236
```

```
PAGE 0029                                  VPAS
                              CROSS-REFERENCE SYMBOL TABLE
TBL0     00683      00254

TBL90    00682      00263

TCHAR    00372      00367

TEMP     00686      00220      00227      00240      00250      00253      00335
         00346      00350      00353      00405      00409      00410      00415
         00454      00458      00470      00475      00541      00552      00556
         00570      00572      00692

TOBT     00327      00352      00374      00430      00491

VECT     00065      00060      00070      00136

VECT1    00077      00073

VECT2    00111      00097

VECT3    00137      00127

VECT4    00184      00145

VECT5    00158      00151

VECT6    00173      00159

VECT7    00193      00189

VECT8    00198      00194

VPAS     00023      00016      00613      00649      01078

VT56C    00170      00183

WHOLE    00322      00315      00317

X1       00659      00083      00091      00115

X2       00660      00087      00090      00111

XY       00154      00192

Y1       00661      00085      00121      00206      00267      00274      00278

Y2       00662      00089      00124      00208      00212      00216      00268
         00273

ZOPB     00047      00603
```

```
/PLOT3 T=00004 IS ON CR00300 USING 00001 BLKS R=0004

0001  :RP,PLOT3
0002  :RP,MERGE
0003  :RP,PLOT
0004  :TR
```

```
\PLOT3 T=00004 IS ON CR00300 USING 00001 BLKS R=0004

0001   :OF,PLOT3
0002   :OF,MERGE
0003   :OF,PLOT
0004   :TR
```