5105-141

Solar Thermal Power Systems Project
Parabolic Dish Systems Development

DOE/JPL-1060-78

Distribution Category UC-62

Software Used with the Flux Mapper at the Solar Parabolic Dish Test Site

C. Miyazono

September 15, 1984

Prepared for

U.S. Department of Energy

Through an Agreement with
National Aeronautics and Space Administration

by

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL Publication 84-76

5105-141
Solar Thermal Power Systems Project
Parabolic Dish Systems Development

DOE/JPL-1060-78
Distribution Category UC-62

Software Used with the Flux Mapper at the Solar Parabolic Dish Test Site

C. Miyazono

September 15, 1984

Prepared for
U.S. Department of Energy
Through an Agreement with
National Aeronautics and Space Administration
by
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL Publication 84-76

Prepared by the Jet Propulsion Laboratory, California Institute of Technology, for the U.S. Department of Energy through an agreement with the National Aeronautics and Space Administration.

The JPL Solar Thermal Power Systems Project is sponsored by the U.S. Department of Energy and is part of the Solar Thermal Program to develop low-cost solar thermal and electric power plants.

ABSTRACT

Software for data archiving and data display was developed for use on a Digital Equipment Corporation (DEC) PDP-11/34A minicomputer for use with the JPL-designed flux mapper. The flux mapper is a two-dimensional, high radiant energy scanning device designed to measure radiant flux energies expected at the focal point of solar parabolic dish concentrators. Interfacing to the DEC equipment was accomplished by standard RS-232C serial lines. The design of the software was dictated by design constraints of the flux-mapper controller. Early attempts at data acquisition from the flux-mapper controller were not without difficulty. Time and personnel limitations resulted in an alternative method of data recording at the test site with subsequent analysis accomplished at a data evaluation location at some later time. Software for plotting was also written to better visualize the flux patterns. Recommendations for future or alternative development are discussed. A listing of the programs used in the analysis is included in an appendix.

ACKNOWLEDGMENT

The work described herein was conducted by the Jet Propulsion Laboratory, California Institute of Technology, for the U.S. Department of Energy through an agreement with the National Aeronautics and Space Administration (NASA Task RE-152, Amendment 327; DOE/ALO/NASA Interagency Agreement No. DE-AM04-80AL13137).

CONTENTS

<u>Table</u>

SECTION I

INTRODUCTION

A. OVERALL DESIGN PHILOSOPHY

The flux mapper is a three-dimensional scanning system to measure the high radiant flux levels expected at the focal point of a solar parabolic dish system. The scanning, measurement, and initial storage of the data are handled by the flux-mapper controller. Software was written to enable a Digital Equipment Corporation (DEC) PDP-11/34A minicomputer to archive the data for long-term storage and to display the data collected from the flux-mapper controller in different formats.

The overriding system requirement for the software was compatibility with the output of the flux-mapper controller. The design, fabrication, and software control of the flux mapper were carried out by JPL personnel. The flux-mapper controller is a microprocessor-based system using read-only memory (ROM) to store the acquisition and output routines. Consequently, changes in the software routines from the flux-mapper controller are more difficult to achieve. It was decided that the requirement for compatibility would rest with the PDP-11/34A because the programming was to be done using standard Fortran IV language and compiler. Therefore, changes in gathering of the data from the controller could be made quickly.

Another important design requirement was ease of use in the field. The flux mapper was to be used at the Parabolic Dish Test Site (PDTS) located at the JPL Edwards Test Station (ETS) in the high desert, 120 km northeast of JPL (in Pasadena). The personnel at the PDTS were not extensively trained in minicomputer programming. Also, the small size of the staff precluded the availability of an individual dedicated to the minicomputer system. Therefore, the staff at the PDTS did not have the time nor the expertise to perform significant minicomputer tasks.

Finally, it was felt that error checking of the incoming data from the flux-mapper controller should be included. Incorrect characters would obviously be detrimental to output displays and listings as well as to the archived data. Because the PDP-11/34A minicomputer would receive data from the flux-mapper controller, error checking could only be done as the data were received. This precludes the use of standard methods such as checksums; therefore, a different approach had to be used.

The three above-mentioned overall design requirements were the basis by which all software was designed. These initial decisions were made during the end of 1979 when discussions were first taking place regarding the flux mapper and its interface with the data-acquisition minicomputer system at the PDTS.

B. DATA-ACQUISITION HARDWARE

The data-acquisition hardware available at the PDTS consisted of a Digital Equipment Corporation PDP-11/34A minicomputer with two removable RK05 disk drives and 256 kilobytes of internal memory. The system also had as supporting peripherals a Kennedy Model 9100 magnetic tape drive, a Versatec Model 1100 printer/plotter, and a Control Data Corporation Model 9766 removable disk drive. In addition, there were interfaces for the various terminals used as well as additional serial ports for access by other devices such as the data loggers. The entire minicomputer system was housed in a mobile trailer located adjacent to the PDTS control room at the test site. All connections between the minicomputer, the terminals, and data loggers (with the exception of the console terminal) were by RS-232C standard serial interfaces. The console terminal communicated by the standard 20-milliamp current loop.

C. GENERAL PRACTICAL RESTRICTIONS

A significant restriction that was not expected occurred as a result of the operating system used. The multi-user system supplied by Digital Equipment Corporation (DEC) called RSX-11M was used during the entire period of development. Unlike the DEC single-user system, the RT-11, their multi-user system would not permit the use of a ring buffer to store input data. During each cycle, the operating system polls all input devices to determine if a task has been initiated. During this polling procedure, inputs to other ports are not placed into ring buffers for temporary storage until they are polled in turn. This resulted in the possibility of losing input characters from a device while the operating system was polling other devices.

With this problem in mind, it became clear that during transfer of data from the flux-mapper controller to the PDP-11/34A the possibility of data loss would be great if this polling mechanism were left intact. This could cause a significant restriction of implementation of the data-acquisition and archiving software.

SECTION II

INITIAL DATA-LOGGING ATTEMPTS

A. DESIGN PHILOSOPHY

During the initial phases of the development of the necessary software, the same general design criteria stated previously were used. Compatibility, ease of use, and error checking were of foremost importance in the design of the software.

During software development, accessibility of the flux mapper for testing was limited. Therefore, much of the development of the software had to be done without access to the actual hardware. This necessitated the inclusion of ease of hardware interfacing as a design criterion. And finally, because the equipment was to be used at the PDTS, where the personnel did not have minicomputer or microprocessor expertise, minimal interfacing in terms of training and operator instructions was a desirable criterion.

B. AVAILABLE HARDWARE

The available hardware was discussed previously. The flux-mapper controller output, basically an RS-232C serial interface line, was connected to one of the serial inputs of the minicomputer.

C. SOFTWARE WRITTEN

The initial software written consisted of a data-acquisition program and a data-output program. Both were written in Fortran IV for the DEC PDP-11/34A minicomputer using the RSX-11M operating system.

The data-acquisition software was written to read the seven-bit ASCII characters that are transmitted from the flux-mapper controller. The format for the data from the flux-mapper controller consists of three types of records: a header record, a set of data records, and an end record.

Each record was identified by a one-character ASCII identifier, followed by the data in a fixed structure. The structures of each type of record are given in Table 1. Note that there was only one header record and only one end record, but there could be many data records.

Each data record represented one traverse of the radiometer probe across the flux mapper. At the end of the traverse, the probe incremented in the perpendicular direction by a preset amount and continued its traverse in the opposite direction. This boustrophedonic motion is evident in the data records as alternating signs on the X spacing entry.

Table 1. Data File Structure

Header Record, Once/Scan	
Header Record Code "H"	A1
Metric/English "M" or "E"	A1
Probe Calibration	A6
Module ID	A2
Test Number	I4
Run ID	I2
Scan ID Number	I3
Software Update Number	I4
Hour	I2
Minute	I2
Second	I2
Month	I2
Day	I2
Year	I2
Probe Type	I1
Scan Type "1" = Rectilinear	I1
Channel 0 Amp Ratio Code	I1
Channel 1 Amp Ratio Code	I1
Channel 2 Amp Ratio Code	I1
Channel 3 Amp Ratio Code	I1
Scale Factor Code	I1
Number of Raster Repeats	I2
Spacing X	F6.2
Raster X Delta	F6.2
Raster Y Delta	F6.2
Initial X Position	F6.2
Initial Y Position	F6.2
Initial Z Position	F6.2
Zero X Position	F6.2
Zero Y Position	F6.2
Zero Z Position	F6.2

Table 1. Data File Structure (Cont'd)

Header Record, Once/Scan	
Zero Data Value	F6.2
Reference Intensity Value	F6.2
Local Intensity Value	F7.2
Data Record, Once/Line of Scan	
Data Record Code "D"	A1
Number of Data Points	I3
Hour	I2
Minute	I2
Second	I2
Spacing X and Direction (sign)	F6.2
Reference Intensity at Time HH:MM:SS	F6.2
X Position of Data Point 1	F6.2
Y Position of Data Point 1	F6.2
Data Point 1	F6.2
Data Point 2, etc.	F6.2
End Record, Once/Scan	
End of Scan Code "E"	A1
Reference Intensity	F6.2
Check Sum of Absolute Value of All Data Points	F12.2
Summation Value (Negative = Overflow)	I9
Points Summed	I4
Number of Lines	I3
End Hour	I2
End Minute	I2
End Second	I2
End of Scan Character "!"	A1

All entries in all the types of records were separated by a comma. All numerical entries were ASCII characters, with or without a leading positive or negative sign. All floating point numbers required a decimal point in the entry while integers did not. The only character that was not from the above list was the last character of the scan. This character was an "!", ASCII code octal number 41.

The acquisition software accessed the data port and verified the character type and structure of each record. If both were correct, then the information was decoded from ASCII to binary numbers when appropriate and stored on magnetic tape in a binary data file. This procedure was to be accomplished at the end of each complete raster of the flux mapper. At that time, the flux-mapper memory would be initialized and the next raster would be taken.

The initial version of the data-output program was designed to read the binary data file on magnetic tape and print out the numbers as they were gathered from the flux mapper. No initial processing other than formatting of the data was to take place.

D. PROBLEMS ENCOUNTERED

Several immediate problems were encountered using these initial versions of the software, including polling with the operating system and transmission. The latter problem was never solved.

As mentioned previously, the polling problem with the operating system placed a severe restriction on data acquisition. During the polling of all serial inputs, the operating system stores characters in a buffer for only the serial input being polled. The other inputs are ignored. The flux-mapper controller transmitted data to the serial port at a regular pattern regardless of the status lines in the RS-232C cable. This regular transmission pattern occasionally overlapped with the operating system's polling of the other ports. The result was that the character transmitted at that time was lost. To overcome this problem, the priority of the data acquisition task was altered.

The RSX-11M operating system features a series of priority levels at which tasks can be assigned. All tasks with the same priority are polled in a round-robin fashion, and all of the users with the same priority level are given an equal opportunity to use the resources of the central processor unit (CPU). Normally, tasks are given a priority level of 50 out of a maximum of 250. Some tasks that require more of the CPU's resources are assigned higher priorities. An example of this is the text editor. It is normally installed at a priority of 65 because its interactive nature requires more of the CPU's resources.

The data-acquisition program required all of the CPU's resources for recording transmitted data from the flux-mapper controller. The acquisition program was installed at a high priority, 249, prior to execution. At execution, this task occupied virtually all of the CPU's resources and, therefore, gathered all the data without difficulty. However, monopolizing all of the resources essentially rendered the multi-user system a single-user system.

During initial use of the acquisition software, it was found that decode errors were occurring in the data string. These errors occurred randomly in the string, but in each occurrence the program would abnormally exit and abnormally close the binary data file on magnetic tape as a file of zero length. This was particularly annoying when the program would fail and all but the last few scans had been transmitted. In addition, this zero-length file on magnetic tape presented problems of playback of subsequently recorded data and also used a file name for null data.

It was suggested that perhaps the flux-mapper controller had transmitted the incorrect character, causing the program to fail. This was checked by recording the output of the flux-mapper controller onto a digital data cassette tape unit. When several runs of the same set of data were recorded onto the cassette, a direct playback showed that these errors did not occur in every run nor at the same location of the run in which they occurred.

It was decided that, to minimize the procedure for the personnel at the PDTS, the option of the analysis and printout immediately after a test was abandoned. Instead, the acquisition of the data was ensured in a simple way. The method used was the digital data cassette recorder mentioned above. At the end of each raster scan, the PDTS personnel transmitted the data from the flux-mapper controller to the digital data cassette three times. The cassette was then sent to JPL for archiving and printout. The three runs ensured that at least one complete and correct run had been stored.

The digital data cassette recorder presented its own set of problems. In the record mode, the cassette fills a buffer and then transfers the data to tape in a single block. If the recorder was switched from the record mode, the data in a partially filled buffer was not transferred to tape; it was lost. Therefore, the recorder, once set to record mode, was left in record mode until all rasters of all the tests had been recorded for that particular day. The recorder presented a problem when tapes were recorded on one side and then reversed. Data on the first side appeared to be erased. The staff at the PDTS was instructed not to reverse the tapes. With these instructions, the data was archived and printed at JPL at a later time.

Finally, the use of the digital data cassette recorder allowed the PDP-11/34A to be used to acquire data from the Acurex Autodata-Nine data logger during the flux-mapper runs. The data-acquisition program monitored the data for various warnings, such as low cooling water flow, and displayed this information on a monitor. The data-acquisition program for the data loggers and for the flux-mapper controller are mutually exclusive tasks because of the nature of the operating system. It was felt that the warning alarms were an additional bonus.

SECTION III

LOGGING METHOD USED

A. DESIGN PHILOSOPHY

The software was designed with three general design criteria in mind: compatibility with the flux-mapper controller, ease of use in the field, and error-checking capability. A large trade-off resulted for the criterion of ease-of-field use; thus, processing the data was made much more difficult.

B. AVAILABLE HARDWARE

The digital data cassette recorders, originally purchased as a backup unit for the data loggers, proved to be the important link between the flux-mapper controller and the PDP-11/34A. Two recorders, one at the test site for the recording of data and one at JPL for playback of data, were available. The one at the test site was set to the output characteristics of the flux mapper. The unit at JPL for data replay was set to an available port. The tapes used were cassette tapes of digital computer quality that contained archived as well as processed data.

C. SOFTWARE WRITTEN

The use of the digital data cassette recorders altered the original software tasks. The software tasks were subsequently divided into three parts: (1) software to read the data from cassette tape and write onto a disk storage medium, (2) software to read from the disk storage medium and transfer to a nine-track magnetic tape for archiving, and (3) software to print out the results from magnetic tape.

The program to read the data from cassette tape and place it onto disk storage media was called CASTAP.FTN. This program had to be an installed task with logical unit number 3 reassigned to the input port. This task also required a high priority to bypass the polling option. (See the RSX-11M V.3.1 operator's manual for details.)

The program CASTAP transferred a given raster scan from the data cassette to a file on disk. The actual ASCII characters were transferred -- no conversion of any type was made on the data elements. The program allowed the entry of the disk file name and allowed the selection of the scan to be stored on disk. The scan numbering system started with the present scan and incremented each time that the end of scan character, the "!", was found. At the conclusion of the data transfer to disk, the cassette recorder was turned off by the program.

The next step was to visually check the data file for incorrect characters. This was a rather poor method of error checking, but considering the limits of time and personnel available, it was the only one possible. The most common extraneous characters found in a record were lower-case characters and carriage returns embedded in data records. On finding errors, one of two options was available. The entire scan could be rerecorded, using one of the other three scans recorded; or a visual inspection of the data cassette rasters, using the data cassette recorder playing back directly into a terminal, would display the data from one of the other scans. The data in the disk file could then be changed to the correct value by using the text editor. To ensure that the file was examined, a comma, ",", had to be added to the end of the disk data file following the end of raster character, the "!". If this comma were not added, then the file was deemed incomplete, and the archiving program would abort abnormally.

The program to transfer the data from disk to a binary file on nine-track magnetic tape was called FMPCAS.FTN. This program read the ASCII file on disk, checked the record structure, decoded the ASCII to binary, and stored the binary in a nine-track magnetic tape to be initialized, if new, and be software-mounted. The magnetic tape file was opened as a Fortran logical unit and interfaced with the operating system.

Once the program had been transferred to magnetic tape, the data could be printed out onto the line printer using the program FMPRINT.FTN. This program required the input of the data file name as well as requiring that the magnetic tape be mounted. Each file had to be called separately for printing; however, several copies could be produced with each call of this program.

D. PROBLEMS ENCOUNTERED

The problems with the software were quite evident. The delay between acquisition at the PDTS and the final printout, the slowness of the acquisition itself, and the tedious data scanning using the editor were all problems that would have been attacked, had there been time and personnel available. Twice a year, perhaps, rasters were taken over the course of a two-week period. This operation, therefore, was relatively infrequent and not commanding priority of time and personnel.

E. SOFTWARE UPDATES

From this version of the acquisition software, a major alteration to the data format was executed by the flux-mapper-controller programming group. Engineers analyzing the flux-mapper data requested time information for each scan. Because a typical complete raster would normally take from one-half hour to one-and-one-half hours, it was felt that these data would be very important in correlating the flux-mapper intensities to weather data such as insolation. The programmers added time information at the beginning of each data record. In addition, the output order was changed to its present boustrophedonic form. Both these changes occurred in June 1981. The software used for data acquisition included these changes. The plotting software (see below) included both formats.

SECTION IV

DISPLAY SOFTWARE

A. DESIGN PHILOSOPHY

It was decided that a three-dimensional plot of the flux-mapper information would be the best way to represent the data for quick review. (See Appendix B.) It was felt that the plots should be easily understandable. Flexibility in plotting was also considered an important criterion. Viewing the flux map from various angles would greatly aid in understanding the resulting patterns. The plots were designed to provide the data in a uniform manner to allow easy comparison between rasters of different distances from the focal plane.

B. AVAILABLE HARDWARE

The same basic minicomputer hardware was available. For printing and plotting, a Versatec Model 1100 electrostatic printer/plotter was used. The associated software to interact with the plotter portion of the device was purchased also from Versatec specifically for RSX-11M version 3.1.

C. SOFTWARE WRITTEN

The display software was a set of programs to input, reformat, calculate, and plot the flux-mapper raster data into a three-dimensional plot viewed from any location. The software was modeled after a plotting package in use on the Univac 1100 computer at the time of this development. Because of the size of the task involved, three separate major programs were written, along with several other minor programs.

The first minor program, TPDK.FTN, read the data from the nine-track magnetic tape and transferred it to a general file on disk in the appropriate format. The same general file name on disk was used each time the program was called. Because the program accessed the tape file as a Fortran logical unit, the magnetic tape had to be software-mounted.

The first major program, GPLOT1D.FTN, read the data from the disk file and reformatted it for use by the other plotting routines. This routine also determined the type of plot desired, such as which view and the presence of contour lines.

The second major program, GPLOT2D.FTN, performed the actual three-dimensional calculations, including the hidden line algorithm. The results were then placed in a file for final plotting in the last phase.

The third major program, PLOTERD.FTN, took the output from the previous file and created the Versatec plotter file. In this program, all of the interfacing to the Versatec software was included. A short, 30-character title was also requested as well as changes in scale factor. The resulting files, VECTR1.BIN and PARM.BIN, contain the plotting instructions for the Versatec plotter.

The last step was to invoke the plotter, using the Versatec-supplied program, RASM.TSK. This program accessed VECTR1.BIN and PARM.BIN to produce the actual plots.

A set of modified versions of the major plotting programs was created for use at the PDTS. The number of optional features, such as viewing location, were fixed. These were denoted as FPLOT1.FTN, FPLOT2.FTN, and PLOTER.FTN.

A second set of modified versions was created to view the plot from directly overhead, namely, a contour plot showing isointensity lines of solar radiation. The same general steps were followed in this set of programs, named CPLOT.FTN, CDRIVE.FTN, and SUMRAD.FTN.

These plots, combined with the data printouts, provided a complete picture of the solar irradiation.

SECTION V

RECOMMENDATIONS AND CONCLUSIONS

A. SOFTWARE RECOMMENDATIONS

The software recommendations discussed here would have been implemented, given sufficient time and personnel. Because of the software's infrequent use, most of the changes and modifications were made either just prior to or immediately after its use, when its priority was high.

The plots and data printouts were analyzed and used by JPL personnel. The filing system that was used did not match the one used by the data-acquisition software. Much of the correlation of older data has been from descriptive information included on the plots and printouts. It would have been very useful to provide more space for descriptive information on both the plots and printouts.

The operating system and the problem with the system polling may have required some system programming modification. At the time, no available personnel had the expertise to examine the problem. Also, the operating system was an older, unsupported version. The newer versions may, in fact, alleviate or mitigate this buffer/polling problem.

The tedius work of going through the disk file after transfer from the digital data cassette using the text editor might have been alleviated by a program that would scan the ASCII data for inappropriate characters. Time did not permit the writing of this program.

Finally, with the correct type of hardware, it should be possible to reproduce the three-dimensional flux-mapper plots on a video terminal equipped for graphics output. With interactive features, this would allow for easier interpretation of data plots.

B. HARDWARE RECOMMENDATIONS

During the initial checkout of the software, the unavailability of the flux-mapper hardware made correction and modification difficult. It would have been helpful if a hardware simulator had been available. An alternative would have been to perform all of the development work, both hardware and software, in one location.

The Versatec printer/plotter was an older model, no longer in production. It normally required several minutes to produce a plot. A faster plotter would have been useful.

C. CONCLUSIONS

The flux-mapper software, as the flux mapper itself, was a laboratory tool. Although the use of the software was cumbersome at times, it consistently provided useful plots and printouts to evaluate the performance of point-focusing parabolic dishes.

APPENDIX A

LIST OF COMPUTER PROGRAMS

This Appendix contains one of the representative sets of programs used to analyze and plot the flux-mapper data. The set included herein is the standard set used for analysis of most flux-mapper runs. This software was available to the PDTS staff; analysis of data contained on digital cassettes was usually performed at the main JPL facility.

The programs included in this representative set of software are

FMTPDK.FTN	Flux-mapper mag tape to disk program
FPLOT1.FTN	First stage of the field three-dimensional plot system
SET32.FTN	Subroutine for three-space to two-space transformation
CLSET.FTN	Subroutine to calculate contour values
FPLOT2.FTN	Second stage of the field three-dimensional plot system
SEG.FTN	Subroutine to write plotting data into a file
CTCELL.FTN	Subroutine to compute contour lines
DRAW.FTN	Subroutine to draw visible part of a line between two points
PLOTER.FTN	Third stage of the field three-dimensional plot system

All of the software generated for the flux-mapper task is not included here due to space constraints. Copies of this software were transferred to Sandia National Laboratories-Albuquerque (SNLA) and are available through that organization.

Major software packages developed for the flux mapper but not listed in this Appendix include the following:

CPLOT.CMD	Command file to run contour plot software system
CPLOT.FTN	First stage of contour plot system
CPLOTO.FTN	First stage of contour plot system for old flux-mapper files
CDRIVE.FTN	Second stage of contour plot system
GPLOT.CMD	Command file to run generalized plot software system
GPLOT1D.FTN	First stage of generalized three-dimensional plot system
GPLOT2D.FTN	Second stage of generalized three-dimensional plot system
PLOTERD.FTN	Third stage of generalized three-dimensional plot system
SUMEVN.FTN	Superimposes two flux-mapper files of unequal size
SUMPRT.FTN	Prints flux-mapper data to printer
SUMRAD.FTN	Integrates flux over a user-defined area
SUMRADO.FTN	Integrates flux over a user-defined area for old flux-mapper files
SUMUP.FTN	Adds the contents of two flux-maper files together

All of the above software was available for use by the PDTS staff. However, only the standard field packages included in this Appendix were consistently used.

```
C
C       FMTPDK.001
C
C       Flux mapper tape to disk program.
C       This program is designed to
C       dump the contents of the flux mapper
C       tape onto disk for plotting
C       and other uses.
C
C       JPL PFDRT. Written by Stephen Ritchie
C
C       .001  15-SEP-80 INITIAL VERSION
C
C       LINK:  Uses standard libraries
C
C**********************************************
        LOGICAL*1 IY,YY,ICD,FILE(14),BK(14),IN,
     * NE
        DIMENSION DTA(64),ISCOM(41),RDTA(64)
        INTEGER TM(3),Z,Y,HDR(9),STYPE,DAY,YEAR,RID,TPROBE,PCAL(3),SID,
     *END,SUNO,CSF(4),SFC,TNO
        REAL LIV
        DATA YY/'Y'/
        DATA FILE/'M','T','0',':','F','M','0','0','0','0','.','D',
     *'A','T'/
        DATA IHR/'FH'/,IDR/'FD'/,ITR/'FE'/
        ITAPE = 4
        TYPE *,' Enter Flux mapper file name in XXXX.YYY format'
        ACCEPT 902,(FILE(I),I=5,14)
C       TYPE 903,FILE
        CALL ASSIGN (ITAPE,FILE,14)
        REWIND ITAPE
        NO = 1
        OPEN (UNIT=1,NAME='TEMP.DAP',TYPE='NEW',FORM='UNFORMATTED',
     *ACCESS='DIRECT',ASSOCIATEVARIABLE=NO,RECORDSIZE=100,DISP='SAVE')
        READ (ITAPE)
        READ (ITAPE)
        READ (ITAPE)
        READ (ITAPE)
        READ (ITAPE)(ISCOM(I),I=1,10)
        WRITE (1'NO) (ISCOM(I),I=1,10)
        READ (ITAPE)IH
        IF (IH.EQ.IHR) GO TO 20
        TYPE 905,IH
        GO TO 50
 20     READ (ITAPE)METRIC,PCAL,MODID,TNO,RID,SID,SUNO,TM,MONTH,DAY,
     *YEAR,TPROBE,STYPE,CSF,SFC,NRR,SX,DX,DY,XI,YI,ZI,ZX,ZY,ZZ,ZDV,RIV,
     *LIV
        WRITE (1'NO)IH,METRIC,PCAL,MODID,TNO,RID,SID,SUNO,TM,MONTH,DAY,
     *YEAR,TPROBE,STYPE,CSF,SFC,NRR,SX,DX,DY,XI,YI,ZI,ZX,ZY,ZZ,ZDV,RIV,
     *LIV
 23     READ (ITAPE)ID
        IF (ID.EQ.IDR) GO TO 25
        IF (ID.EQ.ITR) GO TO 29
        TYPE 910,ID
```

```
      GO TO 50
 25   READ (ITAPE) N,I1,I2,I3,(DTA(I),I=1,N)
C     IF (DTA(1).GE.0.0) GO TO 27
      GO TO 27
C     RE-ORDERING INPUT DATA
      DO 26 I=5,N
      JJ = 65 - I
      RDTA(JJ) = DTA(I)
 26   CONTINUE
      WRITE(1'NO) ID,N,dta(1),dta(2),dta(3),dta(4),(RDTA(I),I=JJ,60)
      GO TO 28
 27   WRITE(1'NO) ID,N,(DTA(I),I=1,N)
 28   CONTINUE
      GO TO 23
 29   CONTINUE
      READ(ITAPE)RIN,SCS,SV,NPS,NLINES,IER,IEM,IES
      WRITE (1'NO) ID,RIN,SCS,SV,NPS,NLINES,IER,IER,IER
      CLOSE (UNIT=1)
      CALL CLOSE (ITAPE)
 50   STOP
 901  FORMAT (A1)
 902  FORMAT (20A1)
 903  FORMAT (1H 10010)
 904  FORMAT (1H F7.2)
 905  FORMAT (' HEADER RECORD DOES NOT MATCH',A2)
 910  FORMAT (' DATA RECORD DOES NOT MATCH',A2)
      END
```

2)

```
      PROGRAM FPLOT1
      LOGICAL*1 YY,DILE(10),ODILE(8),IY,BELL
      DIMENSION X(33) ,Y(33)    ,Z(1089)   ,M(2178) ,
     *         S(6) ,DTA(40) ,Y1(33)
      DIMENSION MXS(2)      ,MXF(2)  ,MXJ(2)   ,MYS(2)   ,MYF(2)   ,MYJ(

      COMMON /SRFBLK/ LIMU(1024),LIML(1024)    ,CL(41)    ,NCL ,
     *             LL      ,FACT        ,IROT      ,NDRZ      ,
     *             NUPPER        ,NRSWT ,BIGD      ,UMIN      ,
     *             UMAX    ,VMIN        ,VMAX      ,RZERO     ,
     *             NOFFP         ,NSPVAL      ,SPV ,BIGEST
      COMMON /PWRZ1/  XXMIN     ,XXMAX      ,YYMIN      ,YYMAX     ,
     *             ZZMIN      ,ZZMAX    ,DELCRT    ,EYEX      ,
     *             EYEY ,EYEZ
      COMMON /SET/CVAL(10) ,NCHAR    ,IHEAD(10),XDIST,NON      ,NVAL
      COMMON /SETN/ NN
      DATA BELL/"7/
      DATA SPVAL,IOFFP/0.0,0/,YY/'Y'/
      DATA STEREO /0.0/
      DATA IFR,ISTP,IROTS,IDRX,IDRY,IDRZ,IUPPER,ISKIRT,NCLA/
     *       1,   0,     0,   1,   1,    0,      0,     0,   6/
      DATA THETA,HSKIRT,CHI,CLO,CINC/
     *      .02,    0.,  0.,  0.,   0./
      DATA S/40.,10.,10.,0.,0.,0./MDZ,NX,NY,NCHAR/33,33,33,10/
      NRSWT = 0
      IEND = -9999
      ISPVAL = -999
      BIGEST = 1.E37
      END = -9999.
      NO = 1
      TYPE *,' THIS PROGRAM PERFORMS A 3-D PLOT OF FLUX MAPPER DATA'
      OPEN(UNIT=4,NAME='TEMP.DAP',TYPE='OLD',FORM='UNFORMATTED',
      1ACCESS='DIRECT',ASSOCIATEVARIABLE=NO,RECORDSIZE=100)
905   FORMAT (10A1)
906   FORMAT (8A1)
30    CONTINUE
      TYPE *,' Enter number of X points in scan, INTEGER'
      ACCEPT 911,NX
      TYPE *,' Enter number of Y lines in scan, INTEGER'
      ACCEPT 911,NY
911   FORMAT (I5)
907   FORMAT (10A2)
      TYPE *,' Do you want Front(1) or Side(2) view? Enter 1 or
     * 2  INTEGER'
      ACCEPT 911,NV
      IF (NV.EQ.1) GO TO 304
      S(1) = 10.
      S(2) = 40.
304   CONTINUE
908   FORMAT (F7.2)
305   TYPE *, ' Do you want contours? Y(es) or N(o)'
      ACCEPT 905,IY
      IF (IY.GT.8288) IY = IY - 32
      IF (IY.EQ.YY) IDRZ = 1
      READ (4'NO)
      READ (4'NO)IH,MM,M1,M2,M3,M4,M5,M6,M7,M8,M9,N,N1,N2,N3,N4,N5,N6,
```

```
     * N7,N8,N9,I1,I2,I3,DDX,X1,X2,X(1),X3,X4,X5,X6,X7,X8,X9,X10
       IF (I2.EQ.0) AIJ = .0125
       IF (I2.EQ.1) AIJ = .5
       IF (I2.EQ.2) AIJ = 1.
       IF (I2.EQ.3) AIJ = 2.
       IF (I2.EQ.4) AIJ = 5.
       IF (I2.EQ.5) AIJ = 10.
       IF (I2.EQ.6) AIJ = 20.
       IF (I2.EQ.7) AIJ = 50.
       IF (I2.EQ.8) AIJ = 100.
       IF (I2.EQ.9) AIJ = 200.
       DO 31 I=2,NX
       X(I) = X(I-1) + DDX
31     CONTINUE
       X1G = X(1)
       X2G = X(NX)
       KSWT = 0
       KFLG = 0
       TYPE *,' IS DATA PRE-JUNE 1981? (Y/N)'
       ACCEPT 905,IY
       IF (IY.GT.98) IY = IY -32
       IF (IY.NE.YY) KFLG = 1
       DO 38 K=1,NY
       READ (4'NO)IC,N,(DTA(I),I=1,N)
       Y1(K)  = DTA(4)
       IF (N.EQ.NX+4) GO TO 32
       KCK = N
       N = NX + 4
       TYPE *,' ***** WARNING, NOT ENOUGH X VALUES *****'
       JKX = KCK - 4
       TYPE 917,JKX
917    FORMAT (' ONLY',I3,' VALUES FOR X')
       TYPE 918,BELL
       TYPE 918,BELL
918    FORMAT (1H A1)
32     CONTINUE
       IF (KFLG.EQ.1) GO TO 34
       DO 33 J=5,N
       Z(J-4+NX*(K-1)) = DTA(J)
       ZCMAX = AMAX1(ZCMAX,DTA(J))
33     CONTINUE
       GO TO 38
34     IF (KSWT.NE.0) GO TO 36
       DO 35 J=5,N
       Z(J-4+NX*(K-1)) = DTA(J)
       ZCMAX = AMAX1(ZCMAX,DTA(J))
35     CONTINUE
       KSWT = 1
       GO TO 38
36     DO 37 J=5,N
       Z(J-4+NX*(K-1)) = DTA(N+5-J)
       ZCMAX = AMAX1(ZCMAX,DTA(N+5-J))
37     CONTINUE
       KSWT = 0
38     CONTINUE
```

```
      IF (KFLG.EQ.1) GO TO 40
      DO 39 J=1,NY
      Y(J) = Y1(NY-J+1)
 39   CONTINUE
      GO TO 42
 40   DO 41 J=1,NY
      Y(J) = Y1(J)
 41   CONTINUE
 42   CONTINUE
      CLOSE (UNIT=4)
C     PRINT 901,X,Y,Z
      Y1G = Y(1)
      Y2G = Y(NY)
      ZCMAX = ZCMAX*AIJ
 901  FORMAT (1H 9E14.7)
      NON = 1
      MMXX = MDZ
      NNXX = NX
      NNYY = NY
      STER = STEREO
      NXP1 = NNXX + 1
      NYP1 = NNYY + 1
      NLA = NCLA
      NSPVAL = ISPVAL
      NOFFP = IOFFP
      SPV = SPVAL
      NDRZ = IDRZ
      IF (IDRZ.NE.0)
     *   CALL CLSET(Z,MMXX,NNXX,NNYY,CHI,CLO,CINC,NLA,40,CL,NCL,
     *              ICNST,NOFFP,SPV,BIGEST)
      IF (IDRZ.NE.0) NDRZ = 1 - ICNST
      STHETA = SIN(STER*THETA)
      CTHETA = COS(STER*THETA)
      RX = S(1) - S(4)
      RY = S(2) - S(5)
      RZ = S(3) - S(6)
      D1 = SQRT(RX*RX+RY*RY+RZ*RZ)
      D2 = SQRT(RX*RX+RY*RY)
      DX = 0.
      DY = 0.
      IF (STEREO.EQ.0.) GO TO 102
      D1 = D1*STEREO*THETA
      IF (D2.GT.0.) GO TO 101
      DX = D1
      GO TO 102
 101  AGL = ATAN(RX/-RY)
      DX = D1*COS(AGL)
      DY = D1*SIN(AGL)
 102  IROT = IROTS
      NPIC = 1
      IF (STER.NE.0.) NPIC = 2
      FACT = 1.
      IF (NRSWT.NE.0) FACT = RZERO/D1
      IF (ISTP.EQ.0.AND.STER.NE.0.) IROT = 1
      XDIST = .5*((1024./102.)-.18*FLOAT(NCHAR))
```

```
      IPIC = 1
            NUPPER = IUPPER
            SIGN1 = IPIC*2 - 3
            EYEX = S(1) + SIGN1*DX
            POIX = S(4) + SIGN1*DX
            EYEY = S(2) + SIGN1*DX
            POIY = S(5) + SIGN1*DX
            EYEZ = S(3)
            POIZ = S(6)
            LL = 0
            CALL SET32(POIX,POIY,POIZ,EYEX,EYEY,EYEZ,1)
            LL = IPIC + 2*ISTP + 3
            IF (STER.EQ.0.) LL = 1
            IF (NRSWT.NE.0) GO TO 107
            XXMIN = X(1)
            XXMAX = X(NNXX)
            YYMIN = Y(1)
            YYMAX = Y(NNYY)
            UMIN = BIGEST
            VMIN = BIGEST
            ZZMIN = BIGEST
            UMAX = - UMIN
            VMAX = -VMIN
            ZZMAX = -ZZMIN
C     PRINT 901,UMIN,UMAX,VMIN,VMAX,ZZMIN,ZZMAX
            DO 104 J=1,NNYY
                 DO 103 I=1,NNXX
                      ZZ = Z(I+NX*(J-1))
                      IF (NOFFP.EQ.1.AND.ZZ.EQ.SPV) GO TO 103
                      ZZMAX = AMAX1(ZZMAX,ZZ)
                      ZZMIN = AMIN1(ZZMIN,ZZ)
                 CALL SET32(X(I),Y(J),Z(I+NX*(J-1)),UT,VT,DU,2)
                      UMAX = AMAX1(UMAX,UT)
                      UMIN = AMIN1(UMIN,UT)
                      VMAX = AMAX1(VMAX,VT)
                      VMIN = AMIN1(VMIN,VT)
 103             CONTINUE
 104        CONTINUE
C     PRINT 901,UMIN,UMAX,VMIN,VMAX,ZZMIN,ZZMAX
            WIDTH = UMAX-UMIN
            HIGHT = VMAX-VMIN
            DIF = .5*(WIDTH-HIGHT)
C     PRINT 901,WIDTH,HIGHT,DIF
            IF (DIF) 105,107,106
 105        UMIN = UMIN + DIF
            UMAX = UMAX - DIF
            GO TO 107
 106        VMIN = VMIN - DIF
            VMAX = VMAX + DIF
 107        CALL SET32(POIX,POIY,POIZ,EYEX,EYEY,EYEZ,1)
C     TYPE 900,NNXX,NNYY
 900  FORMAT (1H 13010)
            DO 109 J=1,NNYY
                 DO 108 I=1,NNXX
                 CALL SET32(X(I),Y(J),Z(I+NNXX*(J-1)),UT,VT,DU,2)
```

```
                   M(2*I-1+2*NNXX*(J-1)) = UT
                   M(2*I+2*NNXX*(J-1)) = VT
C     PRINT 910,I,J,M(1,I,J),I,J,M(2,I,J)
 910  FORMAT (1H 2(2I3,2X,I5,7X))
C     PRINT 900,M(1,I,J),M(2,I,J)
C     PRINT 901,UT,VT
 108               CONTINUE
 109           CONTINUE
               DO 110 K=1,1024
               LIMU(K) = 0
               LIML(K) = 1024
 110           CONTINUE
               NXPASS = 1
               IF (S(1).GE.X(NNXX)) GO TO 113
               IF (S(1).LE.X(1)) GO TO 114
               DO 111 I=2,NNXX
                   LX = I
                   IF (S(1).LE.X(I)) GO TO 112
 111           CONTINUE
 112           MXS(1) = LX-1
               MXJ(1) = -1
               MXF(1) = 1
               MXS(2) = LX
               MXJ(2) = 1
               MXF(2) = NNXX
               NXPASS = 2
               GO TO 115
 113           MXS(1) = NNXX
               MXJ(1) = -1
               MXF(1) = 1
               GO TO 115
 114           MXS(1) = 1
               MXJ(1) = 1
               MXF(1) = NNXX
 115           NYPASS = 1
               IF (S(2).GE.Y(NNYY)) GO TO 118
               IF (S(2).LE.Y(1)) GO TO 119
               DO 116 J=2,NNYY
                   LY = J
                   IF (S(2).LE.(Y(J)+.02)) GO TO 117
 116           CONTINUE
 117           MYS(1) = LY - 1
               MYJ(1) = -1
               MYF(1) = 1
               MYS(2) = LY
               MYJ(2) = 1
               MYF(2) = NNYY
               NYPASS = 2
               GO TO 120
 118           MYS(1) = NNYY
               MYJ(1) = -1
               MYF(1) = 1
               GO TO 120
 119           MYS(1) = 1
               MYJ(1) = 1
```

```
            MYF(1) = NNYY
 120    NN = 1
C       PRINT 901,X,Y
C       PRINT 901,Z
C       PRINT 901,XXMIN,XXMAX,YYMIN,YYMAX,ZZMIN,ZZMAX,DELCRT
C       PRINT 901,EYEX,EYEY,EYEZ,FACT,BIGD,BIGEST,RZERO
C       PRINT 901,UMIN,UMAX,VMIN,VMAX
C       PRINT 900,IHEAD,NCL,LL,IROT,NDRZ,NUPPER,NRSWT,NOFFP,NSPVAL
        OPEN (UNIT=3,NAME='SCRT.DAT',TYPE='NEW',FORM='UNFORMATTED',
     * ACCESS='DIRECT',ASSOCIATEVARIABLE=NN,RECORDSIZE=1090)
        WRITE (3'NN)IHEAD,X,Y,XXMIN,XXMAX,YYMIN,YYMAX,ZZMIN,ZZMAX,
     *DELCRT,EYEX,EYEY,EYEZ,NCL,LL,FACT,IROT,NDRZ,NUPPER,NRSWT,
     *BIGD,UMIN,UMAX,VMIN,VMAX,RZERO,NOFFP,NSPVAL,SPV,BIGEST,CL
     * ,NX,NY,MDZ,X1G,X2G,Y1G,Y2G,ZCMAX,AIJ
C       PRINT 900,LIMU
        WRITE (3'NN)(LIMU(I),I=1,1024)
C       PRINT 900,LIML
        WRITE (3'NN)(LIML(I),I=1,1024)
        WRITE (3'NN)(M(I),I=1,2178)
        WRITE (3'NN)(Z(I),I=1,1089)
C       PRINT 901,RX,RY,HSKIRT,XDIST,CVAL,S,CL
C       PRINT 900,NXPASS,NYPASS,MXS,MXF,MXJ,MYS,MYF,MYJ,IDAY,NCHAR
C       PRINT 900,NON,NVAL
        WRITE (3'NN)NXPASS,NYPASS,MXS,MXF,MXJ,MYS,MYF,MYJ,RX,RY,IDAY,
     *          HSKIRT,NDRZ,XDIST,CVAL,NCHAR,NON,NVAL,S,NNXX,NNYY,MMXX,ISKI
RT
        CALL SET32 (X(1),Y(1),Z(1),UT,VT,DUM,3)
        CLOSE (UNIT=3)
        STOP
        END
```

```
      SUBROUTINE SET32(X,Y,Z,XT,YT,ZT,KFLAG)
C
C     This routine implements the 3-space to 2-space transformation
C     by Kuber, Szabo, and Giulieri, The Perspective Representation
C     of Functions of Two VAriables, J. ACM 15, 2 193-204,1968
C
C X,Y,Z          Are the 3-space coordinates of the intersection of the
C           line of sight and the image plane.  This point can be
C           thought of as the point looked at.
C XT,YT,ZT  Are the 3-space coordinates of the eye position.
C
C KFLAG = 2 arguments
C
C X,Y,Z          Are the 3-space coordinated of a point to be
C           transformed.
C XT, YT    The results of the 3-space to 2-space transformation.
C ZT        Not used.
C
C     If LL(in COMMON)=0, XT and YT are in the same scale and X,Y  Z.
C
C     The variable KFLAG has two possible variables
C           1-compute intersection of line of sight
C           2-transform from 3-space to 2-space
C
C     NOTE!!!!!!!!!!!!
C           The KFLAG=3,4 are special debugging flags and are not
C           part of the plot package.
C
C
      COMMON /PWRZ1/ XXMIN ,XXMAX   ,YYMIN    ,YYMAX    ,
     *               ZZMIN   ,ZZMAX   ,DELCRT   ,EYEX     ,
     *               EYEY    ,EYEZ
      COMMON /SRFBLK/ LIMU(1024)    ,LIML(1024)    ,CL(41)    ,NCL ,
     *          LL          ,FACT          ,IROT      ,NDRZ       ,
     *          NUPPER          ,NRSWT          ,BIGD      ,UMIN       ,
     *          UMAX        ,VMIN          ,VMAX      ,RZERO      ,
     *          NOFFP           ,NSPVAL          ,SPV ,BIGEST
      COMMON /SETN/ NN
      DIMENSION  NLU(7)     ,NRU(7)  ,NBV(7)    ,NTV(7)
C
C     picture corner coordinates for LL=1
C
      DATA NLU(1),NRU(1),NBV(1),NTV(1)/10,1014,10,1014/
C
C     picture corner coordinates for LL=2
C
      DATA NLU(2),NRU(2),NBV(2),NTV(2)/10, 924,50, 964/
C
C     picture corner coordinates for LL=3
C
      DATA NLU(3),NRU(3),NBV(3),NTV(3)/100,1014,50,964/
C
C     picture corner coordinates for LL=4
C
      DATA NLU(4),NRU(4),NBV(4),NTV(4)/10,1014,10,1014/
```

```
C
C      picture corner coordinates for LL=5
C
       DATA NLU(5),NRU(5),NBV(5),NTV(5)/10,1014,10,1014/
C
C      picture corner coordinates for LL=6
C
       DATA NLU(6),NRU(6),NBV(6),NTV(6)/10, 512,256,758/
C
C      picture corner coordinates for LL=7
C
       DATA NLU(7),NRU(7),NBV(7),NTV(7)/512,1014,256,758/
       GO TO (1,2,3,4) KFLAG
 1     JUMP3 = 104
       IF (NOFFP.EQ.1) JUMP3 = 103
       AX = X
       AY = Y
       AZ = Z
       EX = XT
       EY = YT
       EZ = ZT
C
C      As much computation as possible is done during execution of
C      SET32 since the transformation is called many times for each
C      call to SET32.
C
       DX = AX - EX
       DY = AY - EY
       DZ = AZ - EZ
C
C      A more careful computation of direction Cosines.
C
       D = 0.
       T = AMAX1(ABS(DX),ABS(DY),ABS(DZ))
C      PRINT 901,AX,AY,AZ,EX,EY,EZ,DX,DY,DZ,T
C      PRINT 900,NOFFP
 900   FORMAT (1H 120010)
 901   FORMAT (1H 9E14.7)
       IF (T.EQ.0.0) GO TO 30
       R1 = DX/T

       R2 = DY/T
       R3 = DZ/T
       D = SQRT(R1*R1 + R2*R2 + R3*R3)
C
C      If D isn't ZERO......
C
       COSAL = R1/D
       COSBE = R2/D
       COSGA = R3/D
       D = D*T
       GO TO 40
C
C      If D is ZERO, ray has no direction: assign direction down
C      X-axis.
```

```
C
 30     CONTINUE
        COSAL = 1.
        COSBE = 0.
        COSGA = 0.
 40     CONTINUE
        AL = ACOS(COSAL)
        BE = ACOS(COSBE)
        GA = ACOS(COSGA)
        SINGA = SIN(GA)
C       PRINT 901,R1,R2,R3,D,COSAL,COSBE,COSGA,AL,BE,GA,SINGA
C       PRINT 900,LL
        JUMP2 = 110
        IF (LL.EQ.0) GO TO 101
        JUMP2= 108
        DELCRT = NRU(LL)-NLU(LL)
        UO = UMIN
        VO = VMIN
        U1 = NLU(LL)
        V1 = NBV(LL)
        U2 = NRU(LL) - NLU(LL)
        V2 = NTV(LL) - NBV(LL)
        IF(UMAX-UMIN) 52,51,52
 51     U3 = 0.
        GO TO 53
 52     U3 = U2/(UMAX-UMIN)
 53     CONTINUE
        IF (VMAX-VMIN) 55,54,55
 54     V3 = 0.
        GO TO 56
 55     V3 = V2/(VMAX-VMIN)
 56     CONTINUE
        U4 = NRU(LL)
        V4 = NTV(LL)
C       PRINT 901,UO,VO,U1,V1,U2,V2,U3,V3,U4,V4,UMAX,VMAX
C       PRINT 900,NRSWT
        IF (NRSWT.EQ.0) GO TO 101
        UO = -BIGD
        VO = -BIGD
        U3 = U2/(2.*BIGD)
        V3 = V2/(2.*BIGD)
C
C       The 3-space point looked at is transformed inot (0,0) of the
C       2-space.  The 3-space Z axis is transformed into the 2-space Y
C       axis.  If the line of sight is close to parallel to the 3-space
C       Z axis, the 3-space Y axis is chosen (instead of the 3-space
C       Z axis) to be transformed into the 2-space Y axix.
C
 101    IF (SINGA.LT.0.0001) GO TO 102
        R = 1./SINGA
        JUMP = 105
C       PRINT 900,JUMP,JUMP2,JUMP3
        RETURN
 102    SINBE = SIN(BE)
        R = 1./SINBE
```

```
      JUMP = 106
C     PRINT 900,JUMP,JUMP3,JUMP2
      RETURN
C
C     Transformation entry point
C
 2    XX = X
      YY = Y
      ZZ = Z
C     PRINT 901,XX,YY,ZZ
C     PRINT 900,JUMP3
      IF (JUMP3.EQ.104) GO TO 104
 103  IF (ZZ.EQ.SPV) GO TO 109
 104  DENOM = (XX-EX)*COSAL + (YY-EY)*COSBE + (ZZ-EZ)*COSGA
      IF (DENOM.NE.0.0) GO TO 1111
      Q = 1.
      GO TO 50
 1111 Q = D/DENOM
 50   CONTINUE
C     PRINT 901,DENOM,Q
C     PRINT 900,JUMP
      IF (JUMP.EQ.106) GO TO 106
 105  XX = ((EX+Q*(XX-EX)-AX)*COSBE-(EY+Q*(YY-EY)-AY)*COSAL)*R
      YY = (EZ+Q*(ZZ-EZ)-AZ)*R
      GO TO 107
 106  XX = ((EZ+Q*(ZZ-EZ)-AZ)*COSAL-(EX+Q*(XX-EX)-AX)*COSGA)*R
      YY = (EY+Q*(YY-EY)-AY)*R
 107  IF (JUMP2.EQ.110) GO TO 110
 108  XX = AMIN1(U4,AMAX1(U1,U1+U3*(FACT*XX-UO)))
      YY = AMIN1(V4,AMAX1(V1,V1+V3*(FACT*YY-VO)))
      GO TO 110
 109  XX = NSPVAL
      YY = NSPVAL
      GO TO 110
 110  XT = XX
      YT = YY
C     PRINT 901,XT,YT
C     PRINT 900,JUMP2
      RETURN
 3    CONTINUE
      WRITE (3'NN)JUMP,JUMP2,JUMP3,EX,EY,EZ,COSAL,COSBE,COSGA,D,
     * AX,AY,AZ,R,UO,U1,U2,U3,U4,VO,V1,V2,V3,V4
C     PRINT 900,JUMP,JUMP2,JUMP3
C     PRINT 901,EX,EY,EZ,COSAL,COSBE,COSGA,D,AX,AY,AZ,R
C     PRINT 901,UO,U1,U2,U3,U4,VO,V1,V2,V3,V4
      RETURN
 4    CONTINUE
      READ (4'NN)JUMP,JUMP2,JUMP3,EX,EY,EZ,COSAL,COSBE,COSGA,D,
     * AX,AY,AZ,R,UO,U1,U2,U3,U4,VO,V1,V2,V3,V4
C     PRINT 900,JUMP,JUMP2,JUMP3
C     PRINT 901,EX,EY,EZ,COSAL,COSBE,COSGA,D,AX,AY,AZ,R
C     PRINT 901,UO,U1,U2,U3,U4,VO,V1,V2,V3,V4
      RETURN
      END
```

```
      SUBROUTINE CLSET(Z,MDZ,NX,NY,CHI,CLO,CINC,NLA,NLM,CL,NCL,
     *ICNST,NOFFP,SPV,BIGEST)
      COMMON /SET/CVAL(10),NCHAR,IHEAD(10),XDIST,NON,NVAL
      DIMENSION Z(1089),CL(41)
      LOGICAL*1 GOODZ
C
C     CLSET COMPUTES THE VALUES OF THE CONTOUR LEVELS
C     AND PUTS THEM IN
C     CL
C
      GOODZ = .FALSE.
      ICNST = 0
      GLO =CLO
      HA = CHI
      FANC = CINC
      CRAT = FLOAT(NLA)
      NCL = 0
      IF (HA-GLO) 110,120,130
  110 GLO = HA
      HA = CLO
      GO TO 130
C
C     If HA and GLO are not set, set them to the values of BIGEST
C     and -BIGEST respectively
C
  120 GLO = -BIGEST
      HA = -GLO
      GO TO 140
  130 IF (FANC.EQ.0.) FANC = (HA-GLO)/(CRAT-1)
  140 CONTINUE
      DO 150 J=1,NY
            DO 145 I=1,NX
            IF (NOFFP.EQ.1.AND.Z(I+NX*(J-1)).EQ.SPV) GO TO 145
            IF (GOODZ) GO TO 146
            ZMIN = Z(I+NX*(J-1))
            ZMAX = Z(I+NX*(J-1))
            GOODZ = .TRUE.
            GO TO 145
  146       ZMIN = AMIN1(Z(I+NX*(J-1)),ZMIN)
            ZMAX = AMAX1(Z(I+NX*(J-1)),ZMAX)
  145       CONTINUE
  150 CONTINUE
C
C     Check HA and GLO to make sure they fall within the range of Z
C     values being plotted; if not, set HA=ZMAX and/or GLO=ZMIN.
c
      IF (HA.GT.ZMAX) HA = ZMAX
      IF (GLO.LT.ZMIN) GLO = ZMIN
C     PRINT 901,HA,GLO,FANC,ZMIN,ZMAX
  901 FORMAT (1H 9E14.7)
C
C     If contour increment has not been set, compute a 'NICE' value
C     for it.
C
      IF (FANC) 160,170,190
```

```
 160    CRAT = -FANC
 170    FANC = (HA-GLO)/(CRAT-1)
        IF (FANC) 220,220,180
 180    P = 10.**(IFIX(ALOG10(FANC)+500.)-500)
        FANC = AINT(FANC/P)*P
C
C       Recompute 'NICE' values of HA and GLO.
C
        GLO = AINT(GLO/FANC)*FANC
        HA = AINT(HA/FANC)*FANC
C
C       Compute contour levels array.
C
 190    DO 200 K=1,NLM
        CC = GLO + FLOAT(K-1)*FANC
        KK = K
        IF (CC.GT.HA) GO TO 210
        CL(K) = CC
 200    CONTINUE
 210    NCL = KK - 1
C
C       Shave away contour values not strictly between ZMIN and ZMAX.
C
 230    CONTINUE
        IF (NCL.LE.0) GO TO 240
        IF (CL(1).GT.ZMIN) GO TO 270
        IF (NCL.LE.1) GO TO 260
              DO 250 I=2,NCL
 250          CL(I-1) = CL(I)
 260    NCL = NCL - 1
        GO TO 230
 270    CONTINUE
        IF (CL(NCL).GE.ZMAX) GO TO 260
 240    CONTINUE
        NVAL = 10
        IF (NCL.LT.10) NVAL = NCL
        DO 241 I=1,NVAL
        J = NCL - I + 1
        CVAL(I) = CL(J)
 241    CONTINUE
        RETURN
 220    ICNST = 1
        NVAL = 10
        IF (NCL.LT.10) NVAL = NCL
        DO 221 I=1,NVAL
        J = NCL - I + 1
        CVAL(I) = CL(J)
 221    CONTINUE
        RETURN
C
        END
```

```
      PROGRAM FPLOT2
      DIMENSION X(33) ,Y(33)   ,Z(1089)  ,M(2178) ,
     *        S(6) ,DTA(40)
      DIMENSION MXS(2)      ,MXF(2)  ,MXJ(2)    ,MYS(2)    ,MYF(2)     ,MYJ(
2)
      COMMON /SRFBLK/ LIMU(1024),LIML(1024)     ,CL(41)    ,NCL ,
     *            LL       ,FACT        ,IROT       ,NDRZ        ,
     *            NUPPER          ,NRSWT ,BIGD       ,UMIN        ,
     *            UMAX    ,VMIN          ,VMAX       ,RZERO       ,
     *            NOFFP           ,NSPVAL       ,SPV ,BIGEST
      COMMON /PWRZ1/  XXMIN      ,XXMAX      ,YYMIN       ,YYMAX         ,
     *            ZZMIN         ,ZZMAX     ,DELCRT      ,EYEX        ,
     *            EYEY ,EYEZ
      COMMON /SET/CVAL(10) ,NCHAR     ,IHEAD(10),XDIST,NON          ,NVAL
     *, X1G,X2G,Y1G,Y2G,ZCMAX,S,NY,AIJ,NX
      COMMON /SETN/ NN
      DATA SPVAL,IOFFP/0.0,0/
      DATA STEREO /0.0/
      DATA IFR,ISTP,IROTS,IDRX,IDRY,IDRZ,IUPPER,ISKIRT,NCLA/
     *        1,    0,     0,   1,    1,    1,       0,        0,    6/
      DATA THETA,HSKIRT,CHI,CLO,CINC/
     *       .02,    0.,  0.,  0.,   0./
      DATA S/-30.,0.,3.,0.,0.,0./MDZ,NX,NY,NCHAR/33,33,33,10/
      NN = 1
      OPEN(UNIT=4,NAME='SCRT.DAT',TYPE='OLD',FORM='UNFORMATTED',
     * ACCESS='DIRECT',ASSOCIATEVARIABLE=NN,RECORDSIZE=1090)
      READ (4'NN)IHEAD,X,Y,XXMIN,XXMAX,YYMIN,YYMAX,ZZMIN,ZZMAX,
     * DELCRT,EYEX,EYEY,EYEZ,NCL,LL,FACT,IROT,NDRZ,NUPPER,NRSWT,
     * BIGD,UMIN,UMAX,VMIN,VMAX,RZERO,NOFFD,NSPVAL,SPV,BIGEST,CL
     * ,NX,NY,MDZ,X1G,X2G,Y1G,Y2G,ZCMAX,AIJ
      READ (4'NN)(LIMU(I),I=1,1024)
      READ (4'NN)(LIML(I),I=1,1024)
      READ (4'NN)(M(I),I=1,2178)
      READ (4'NN)(Z(I),I=1,1089)
      READ (4'NN)NXPASS,NYPASS,MXS,MXF,MXJ,MYS,MYF,MYJ,RX,RY,IDAY,
     * HSKIRT,NDRZ,XDIST,CVAL,NCHAR,NON,NVAL,S,NNXX,NNYY,MMXX,ISKIRT
      CALL SET32 (X(1),Y(1),Z(1),UT,VT,DU,4)
      CLOSE (UNIT=4)
  901 FORMAT (1H 9E14.7)
  900 FORMAT (1H 13I10)
C     PRINT 901,X,Y
C     PRINT 901,Z
C     PRINT 901,XXMIN,XXMAX,YYMIN,YYMAX,ZZMIN,ZZMAX,DELCRT
C     PRINT 901,EYEX,EYEY,EYEZ,FACT,BIGD,BIGEST,RZERO
C     PRINT 901,UMIN,UMAX,VMIN,VMAX
C     PRINT 900,IHEAD,NCL,LL,IROT,NDRZ,NUPPER,NRSWT,NOFFP,NSPVAL
C     PRINT 900,LIMU
C     PRINT 900,LIML
C     PRINT 901,RX,RY,HSKIRT,XDIST,CVAL,S,CL
C     PRINT 900,NXPASS,NYPASS,MXS,MXF,MXJ,MYS,MYF,MYJ,IDAY,NCHAR
C     PRINT 900,NON,NVAL,NNXX,NNYY,MMXX
      GO TO 102
      DO 51 I=1,NNXX
      DO 50 J=1,8
      K=8+J
      L=16+J
```

```
      N=24+J
C     PRINT 903,I,J,X(I),Y(J),Z(I,J),I,K,X(I),Y(K),Z(I,K),
C    * I,L,X(I),Y(L),Z(I,L),I,N,X(I),Y(N),Z(I,N)
 903  FORMAT (1H 4(2I3,3(2X,F5.2),5X))
50    CONTINUE
      PRINT 905
51    CONTINUE
 905  FORMAT (1H )
      DO 76 I=1,NNXX
      DO 75 J=1,8
      K=8+J
      L=16+J
      N=24+J
C     PRINT 904,I,J,X(I),Y(J),M(1,I,J),I,K,X(I),Y(K),M(1,I,K),
C    *       I,L,X(I),Y(L),M(1,I,L),I,N,X(I),Y(N),M(1,I,N)
 904  FORMAT (1HC 4(2I3,2(2X,F5.2),I5,5X))
75    CONTINUE
      PRINT 905
76    CONTINUE
      DO 101 I=1,NNXX
      DO 100 J=1,8
      K=8+J
      L=16+J
      N=24+J
C     PRINT 904,I,J,X(I),Y(J),M(2,I,J),I,K,X(I),Y(K),M(2,I,K),
C    *       I,L,X(I),Y(L),M(2,I,L),I,N,X(I),Y(N),M(2,I,N)
100   CONTINUE
      PRINT 905
101   CONTINUE
 102  CONTINUE
      NON = 1
      OPEN(UNIT=3,NAME='PLOT.DAT',TYPE='NEW',FORM='UNFORMATTED',
     * ACCESS='DIRECT',ASSOCIATEVARIABLE=NON,RECORDSIZE=10)
 120  IF (NXPASS.EQ.2.AND.NYPASS.EQ.2) GO TO 146
      IF (ISKIRT.EQ.0) GO TO 126
      IN = MXS(1)
      IF = MXF(1)
      JN = MYS(1)
      JF = MYF(1)
      IF (NYPASS.NE.1) GO TO 123
      CALL SET32(X(1),Y(JN),HSKIRT,UX1,VX1,DUMMY,2)
      CALL SET32(X(NNXX),Y(JN),HSKIRT,UX2,VX2,DUMMY,2)
      DO 121 I=1,NNXX
            CALL SET32(X(I),Y(JN),HSKIRT,UX,VX,DUMMY,2)
            MU = UX
            MV = VX
      CALL DRAW(MU,MV,M(2*I-1+2*NNXX*(JN-1)),M(2*I+2*NNXX*(JN-1)),1)
 121  CONTINUE
      CALL DRAW (IFIX(UN1),IFIX(VX1),IFIX(UX2),IFIX(VX2),3)
      IF (IDRY.NE.0) GO TO 123
      DO 122 I=2,NNXX
      CALL DRAW(M(2*I-3+2*NNXX*(JN-1)),M(2*I-2+2*NNXX*(JN-1)),
     * M(2*I-1+2*NNXX*(JN-1)),M(2*I+2*NNXX*(JN-1)),3)
 122  CONTINUE
 123  IF (NXPASS.NE.1) GO TO 126
```

```
      CALL SET32(X(IN),Y(1),HSKIRT,UY1,VY1,DUMMY,2)
      CALL SET32(X(IN),Y(NNYY),HSKIRT,UY2,VY2,DUMMY,2)
      DO 124 J=1,NNYY
            CALL SET32(X(IN),Y(J),HSKIRT,UY,VY,DUMMY,2)
            MU = UY
            MV = VY
      CALL DRAW(MU,MV,M(2*IN-1+2*NNXX*(J-1)),M(2*IN+2*NNXX*(J-1)),1)
 124  CONTINUE
      CALL DRAW(IFIX(UY1),IFIX(VY1),IFIX(UY2),IFIX(VY2),3)
      IF (IDRX.NE.0) GO TO 126
      DO 125 J = 2,NNYY
      CALL DRAW(M(2*IN-1+2*NNXX*(J-2)),M(2*IN+2*NNXX*(J-2)),
     * M(2*IN-1+2*NNXX*(J-1)),M(2*IN+2*NNXX*(J-1)),3)
 125  CONTINUE
 126  LI = MXJ(1)
      MI = MXS(1) - LI
      NI = IABS(MI-MXF(1))
      LJ = MYJ(1)
      MJ = MYS(1) - LJ
      NJ = IABS(MJ-MYF(1))
      IF (ABS(RX).LE.ABS(RY)) GO TO 133
      IF (ISKIRT.NE.0.OR.NYPASS.NE.1) GO TO 128
      I = MXS(1)
      DO 127 J=2,NNYY
      CALL DRAW(M(2*I-1+2*NNXX*(J-2)),M(2*I+2*NNXX*(J-2)),
     * M(2*I-1+2*NNXX*(J-1)),M(2*I+2*NNXX*(J-1)),2)
 127  CONTINUE
 128  DO 132 II = 1,NNXX
      I = MI + II*LI
      IPLI = I + LI
      IF (NYPASS.EQ.1) GO TO 129
      K = MYS(1)
      L = MYS(2)
C     PRINT 900,NDRZ,II,NI
      IF (IDRX.NE.0)
     * CALL DRAW(M(2*I-1+2*NNXX*(K-1)),M(2*I+2*NNXX*(K-1)),
     * M(2*I-1+2*NNXX*(L-1)),M(2*I+2*NNXX*(L-1)),3)
      IF (NDRZ.NE.0.AND.II.NE.NI)
     * CALL CTCELL(Z,MMXX,NNXX,NNYY,M,MINO(I,L+LI),K)
 129        DO 131 JPASS=1,NYPASS
                 LJ = MYJ(JPASS)
                 MJ = MYS(JPASS) - LJ
                 NJ = IABS(MJ-MYF(JPASS))
                       DO 130 JJ = 1,NJ
                             J = MJ + JJ*LJ
                             JPLJ = J + LJ
                             IF (IDRX.NE.0.AND.JJ.NE.NJ)
     * CALL DRAW(M(2*I-1+2*NNXX*(J-1)),M(2*I+2*NNXX*(J-1)),
     * M(2*I-1+2*NNXX*(JPLJ-1)),M(2*I+2*NNXX*(JPLJ-1)),3)
                             IF (I.NE.MXF(1).AND.IDRY.NE.0)
     * CALL DRAW(M(2*IPLI-1+2*NNXX*(J-1)),M(2*IPLI+2*NNXX*(J-1)),
     * M(2*I-1+2*NNXX*(J-1)),M(2*I+2*NNXX*(J-1)),3)
C     PRINT 900,NDRZ,JJ,NJ,II,NNXX,I,J,LI,LJ
                             IF(NDRZ.NE.0.AND.JJ.NE.NJ.AND.II
     *                       .NE.NNXX)
```

```
     *                        CALL CTCELL(Z,MMXX,NNXX,NNYY,M,
     *                        MINO(I,I+LI),MINO(J,J+LJ))
  130                   CONTINUE
  131         CONTINUE
  132   CONTINUE
        GO TO 140
  133   IF (ISKIRT.NE.0.OR.NXPASS.NE.1) GO TO 135
        J = MYS(1)
        DO 134 I=2,NNXX
        CALL DRAW(M(2*I-3+2*NNXX*(J-1)),M(2*I-2+2*NNXX*(J-1)),
     * M(2*I-1+2*NNXX*(J-1)),M(2*I+2*NNXX*(J-1)),2)
  134   CONTINUE
  135   DO 139 JJ = 1,NNYY
              J = MJ + JJ*LJ
              JPLJ = J + LJ
              IF (NXPASS.EQ.1) GO TO 136
              K = MXS(1)
              L = MXS(2)
              IF (IDRY.NE.0)
     * CALL DRAW(M(2*K-1+2*NNXX*(J-1)),M(2*K+2*NNXX*(J-1)),
     * M(2*L-1+2*NNXX*(J-1)),M(2*L+2*NNXX*(J-1)),2)
              IF (NDRZ.NE.0.AND.JJ.NE.NJ)
     *        CALL CTCELL(Z,MMXX,NNXX,NNYY,M,K,MINO(J,J+LJ))
  136         DO 138 IPASS=1,NXPASS
                    LI = MXJ(IPASS)
                    MI = MXS(IPASS)-LI
                    NI = IABS(MI-MXF(IPASS))
                    DO 137 II=1,NI
                          I = MI + II*LI
                          IPLI = I + LI
                          IF(IDRY.NE.0.AND.II.NE.NI)
     * CALL DRAW(M(2*I-1+2*NNXX*(J-1)),M(2*I+2*NNXX*(J-1)),
     * M(2*IPLI-1+2*NNXX*(J-1)),M(2*IPLI+2*NNXX*(J-1)),3)
                          IF (J.NE.MYF(1).AND.IDRX.NE.0)
     * CALL DRAW(M(2*I-1+2*NNXX*(JPLJ-1)),M(2*I+2*NNXX*(JPLJ-1)),
     * M(2*I-1+2*NNXX*(J-1)),M(2*I+2*NNXX*(J-1)),3)
C       PRINT 900,NDRZ,II,NI,JJ,NNYY
                          IF(NDRZ.NE.0.AND.II.NE.NI.AND.JJ.NE.NNYY)
     *                    CALL CTCELL(Z,MMXX,NNXX,NNYY,M,
     *                    MINO(I,I+LI),MINO(J,J+LJ))
  137               CONTINUE
  138         CONTINUE
  139   CONTINUE
  140   IF (ISKIRT.EQ.0) GO TO 149
        IF (IDRX.NE.0) GO TO 143
        DO 142 IPASS=1,NXPASS
              IF (NXPASS.EQ.2) IF=1+(IPASS-1)*(NNXX-1)
              DO 141 J=2,NNYY
        CALL DRAW(M(2*IF-1+2*NNXX*(J-2)),M(2*IF+2*NNXX*(J-2)),
     * M(2*IF-1+2*NNXX*(J-1)),M(2*IF+2*NNXX*(J-1)),1)
  141         CONTINUE
  142   CONTINUE
  143   IF (IDRY.NE.0) GO TO 149
        DO 145 JPASS=1,NYPASS
              IF(NYPASS.EQ.2) JF=1+(JPASS-1)*(NNYY-1)
```

```
          DO 144 I=2,NNXX
      CALL DRAW(M(2*I-3+2*NNXX*(JF-1)),M(2*I-2+2*NNXX*(JF-1)),
    * M(2*I-1+2*NNXX*(JF-1)),M(2*I+2*NNXX*(JF-1)),1)
144       CONTINUE
145   CONTINUE
      GO TO 149
146   IF (NUPPER.GT.0.AND.S(3).LE.S(6)) GO TO 149
      IF (NUPPER.LE.0.AND.S(3).GT.S(6)) GO TO 149
      NUPPER = 1
      IF (S(3).LE.S(6)) NUPPER = -1
      DO 148 I=1,NNXX
          DO 147 J=1,NNYY
               IF (IDRX.NE.0.AND.J.NE.NNYY)
    * CALL DRAW(M(2*I-1+2*NNXX*(J-1)),M(2*I+2*NNXX*(J-1)),
    * M(2*I-1+2*NNXX*J),M(2*I+2*NNXX*J),1)
               IF (IDRY.NE.0.AND.I.NE.NNXX)
    * CALL DRAW(M(2*I-1+2*NNXX*(J-1)),M(2*I+2*NNXX*(J-1)),
    * M(2*I+1+2*NNXX*(J-1)),M(2*I+2+2*NNXX*(J-1)),1)
               IF(IDRZ.NE.0.AND.I.NE.NNXX.AND.J.NE.NNYY)
    *          CALL CTCELL(Z,MMXX,NNXX,NNYY,M,I,J)
147       CONTINUE
148   CONTINUE
149   IF(STER.EQ.0.) GO TO 153
      IF (ISTP) 151,150,152
150   CONTINUE
151   CONTINUE
      GO TO 154
152   IF(IPIC.NE.2) GO TO 154
153   CONTINUE
154   CONTINUE
      WRITE (3'NON) END,END, IEND
      CLOSE (UNIT=3)
      RETURN
      END
```

```
      SUBROUTINE SEG(IX1,IY1,IX2,IY2,KFLAG)
C
C     This routine would normally be the interface between the SRFACE
C     package and the plot package.  Because of space limitations in
C     our machine it is used to write a file containing the data to
C     be plotted.  The PLOTER program reads this file and plots the
C     data.
C
C
C
      LOGICAL*1 START,TITLE,CONTUR
      COMMON /SET/CVAL(10) ,NCHAR   ,IHEAD(10)     ,XDIST
     * ,NON ,NVAL,X1G,X2G,Y1G,Y2G,ZCMAX,S,NY,AIJ,NX
      DATA START/.TRUE./,TITLE/.TRUE./,UPERIN/102./,XLAST/0./
      DATA CONTUR/.TRUE./
      INTEGER CHEAD(48)
      DATA CHEAD/'UP',' T','O ','TH','E ','FI','RS','T ','  ','  ',
     * '  ','  ','10',' C','ON','TO','UR',' V','AL','UE','S ','OF',
     * '  ','  ','CO','NS','TA','NT',' Z',' (','HI','GH','ES','T ',
     * '  ','  ','TO',' L','OW','ES','T)','  ','WA','TT','S/','CM',
     *  '**','2 '/
      DIMENSION CVALQ(8),S(6)
C
C     On the first time through this routine the program writes all
C     the headers and labes for the graph.  During subsequent calls the
C     the hidden line data is written for ploting.
C
 5    IF (.NOT.START) GO TO 10
C     TYPE 999,NY,X1G,X2G,Y1G,Y2G,ZCMAX
 999  FORMAT (1H I5,5F12.4)
      XLAST = 0.
      IXTO = 0
      IYTO = 0
      IF (.NOT.CONTUR) GO TO 8
      HT = 8.0
      DO 6 I=1,4
      HT = HT - .5
      II = 12*(I-1)+1
      IK = II + 12
      WRITE (3'NON)HT,(CHEAD(J),J=II,IK),NVAL
 6    CONTINUE
      WRITE (3'NON) X1G,X2G,Y1G,Y2G,ZCMAX
      WRITE (3'NON) S(1),S(2),S(3),NY,NX
 66   HT = 5.5
      DO 7 I=1,NVAL
      HT = HT - .5
C     TYPE 999,I,CVAL(I),AIJ
      CVAL(I) = CVAL(I)*AIJ
      ENCODE(12,77,CVALQ)CVAL(I)
C     TYPE 998,CVALQ
 998  FORMAT (1H 8A2)
 77   FORMAT(F12.1)
      WRITE (3'NON)HT,CVALQ
 7    CONTINUE
      CONTUR = .FALSE.
```

```
8       CONTINUE
        IF (.NOT.TITLE) GO TO 10
        WRITE (3'NON)XDIST,IHEAD
        TITLE = .FALSE.
10      IF (IX1.EQ.IXTO.AND.IY1.EQ.IYTO) GO TO 20
        AX = FLOAT(IX1)/UPERIN
        AY = FLOAT(IY1)/UPERIN
        IP3 = +3
        WRITE (3'NON) AX,AY,IP3
        IF (KFLAG.GT.1) PRINT 901,IPC,AX,AY
901     FORMAT (1H I10,2E14.7)
20      BX = FLOAT(IX2)/UPERIN
        BY = FLOAT(IY2)/UPERIN
        IP2 = +2
        WRITE (3'NON)BX,BY,IP2
        IF (KFLAG.GT.1) PRINT 901,IP2,BX,BY
        IXTO = IX2
        IYTO = IY2
        START = .FALSE.
        XLAST = AMAX1(XLAST,AMAX0(IX1,IX2))
        RETURN
        END
```

```
      SUBROUTINE SET32(X,Y,Z,XT,YT,ZT,KFLAG)
C
C     This routine implements the 3-space to 2-space transformation
C     by Kuber, Szabo, and Giulieri, The Perspective Representation
C     of Functions of Two VAriables, J. ACM 15, 2 193-204,1968
C
C X,Y,Z          Are the 3-space coordinates of the intersection of the
C           line of sight and the image plane.  This point can be
C           thought of as the point looked at.
C XT,YT,ZT  Are the 3-space coordinates of the eye position.
C
C KFLAG = 2 arguments
C
C X,Y,Z          Are the 3-space coordinated of a point to be
C           transformed.
C XT, YT    The results of the 3-space to 2-space transformation.
C ZT        Not used.
C
C     If LL(in COMMON)=0, XT and YT are in the same scale and X,Y  Z.
C
C     The variable KFLAG has two possible variables
C           1-compute intersection of line of sight
C           2-transform from 3-space to 2-space
C
C     NOTE!!!!!!!!!!!!
C           The KFLAG=3,4 are special debugging flags and are not
C           part of the plot package.
C
C
      COMMON /PWRZ1/ XXMIN ,XXMAX    ,YYMIN     ,YYMAX     ,
     *               ZZMIN    ,ZZMAX    ,DELCRT    ,EYEX      ,
     *               EYEY     ,EYEZ
      COMMON /SRFBLK/ LIMU(1024)     ,LIML(1024)      ,CL(41)    ,NCL ,
     *           LL         ,FACT            ,IROT      ,NDRZ      ,
     *           NUPPER           ,NRSWT           ,BIGD      ,UMIN      ,
     *           UMAX       ,VMIN            ,VMAX      ,RZERO     ,
     *           NOFFP            ,NSPVAL          ,SPV ,BIGEST
      COMMON /SETN/ NN
      DIMENSION  NLU(7)     ,NRU(7)   ,NBV(7)    ,NTV(7)
C
C     picture corner coordinates for LL=1
C
      DATA NLU(1),NRU(1),NBV(1),NTV(1)/10,1014,10,1014/
C
C     picture corner coordinates for LL=2
C
      DATA NLU(2),NRU(2),NBV(2),NTV(2)/10, 924,50, 964/
C
C     picture corner coordinates for LL=3
C
      DATA NLU(3),NRU(3),NBV(3),NTV(3)/100,1014,50,964/
C
C     picture corner coordinates for LL=4
C
      DATA NLU(4),NRU(4),NBV(4),NTV(4)/10,1014,10,1014/
```

```
C
C      picture corner coordinates for LL=5
C
       DATA NLU(5),NRU(5),NBV(5),NTV(5)/10,1014,10,1014/
C
C      picture corner coordinates for LL=6
C
       DATA NLU(6),NRU(6),NBV(6),NTV(6)/10, 512,256,758/
C
C      picture corner coordinates for LL=7
C
       DATA NLU(7),NRU(7),NBV(7),NTV(7)/512,1014,256,758/
       GO TO (1,2,3,4) KFLAG
 1     JUMP3 = 104
       IF (NOFFP.EQ.1) JUMP3 = 103
       AX = X
       AY = Y
       AZ = Z
       EX = XT
       EY = YT
       EZ = ZT
C
C      As much computation as possible is done during execution of
C      SET32 since the transformation is called many times for each
C      call to SET32.
C
       DX = AX - EX
       DY = AY - EY
       DZ = AZ - EZ
C
C      A more careful computation of direction Cosines.
C
       D = 0.
       T = AMAX1(ABS(DX),ABS(DY),ABS(DZ))
C      PRINT 901,AX,AY,AZ,EX,EY,EZ,DX,DY,DZ,T
C      PRINT 900,NOFFP
 900   FORMAT (1H 120010)
 901   FORMAT (1H 9E14.7)
       IF (T.EQ.0.0) GO TO 30
       R1 = DX/T

       R2 = DY/T
       R3 = DZ/T
       D = SQRT(R1*R1 + R2*R2 + R3*R3)
C
C      If D isn't ZERO......
C
       COSAL = R1/D
       COSBE = R2/D
       COSGA = R3/D
       D = D*T
       GO TO 40
C
C      If D is ZERO, ray has no direction: assign direction down
C      X-axis.
```

```
C
 30     CONTINUE
        COSAL = 1.
        COSBE = 0.
        COSGA = 0.
 40     CONTINUE
        AL = ACOS(COSAL)
        BE = ACOS(COSBE)
        GA = ACOS(COSGA)
        SINGA = SIN(GA)
C       PRINT 901,R1,R2,R3,D,COSAL,COSBE,COSGA,AL,BE,GA,SINGA
C       PRINT 900,LL
        JUMP2 = 110
        IF (LL.EQ.0) GO TO 101
        JUMP2= 108
        DELCRT = NRU(LL)-NLU(LL)
        UO = UMIN
        VO = VMIN
        U1 = NLU(LL)
        V1 = NBV(LL)
        U2 = NRU(LL) - NLU(LL)
        V2 = NTV(LL) - NBV(LL)
        IF(UMAX-UMIN) 52,51,52
 51     U3 = 0.
        GO TO 53
 52     U3 = U2/(UMAX-UMIN)
 53     CONTINUE
        IF (VMAX-VMIN) 55,54,55
 54     V3 = 0.
        GO TO 56
 55     V3 = V2/(VMAX-VMIN)
 56     CONTINUE
        U4 = NRU(LL)
        V4 = NTV(LL)
C       PRINT 901,UO,VO,U1,V1,U2,V2,U3,V3,U4,V4,UMAX,VMAX
C       PRINT 900,NRSWT
        IF (NRSWT.EQ.0) GO TO 101
        UO = -BIGD
        VO = -BIGD
        U3 = U2/(2.*BIGD)
        V3 = V2/(2.*BIGD)
C
C       The 3-space point looked at is transformed inot (0,0) of the
C       2-space.  The 3-space Z axis is transformed into the 2-space Y
C       axis.  If the line of sight is close to parallel to the 3-space
C       Z axis, the 3-space Y axis is chosen (instead of the 3-space
C       Z axis) to be transformed into the 2-space Y axix.
C
 101    IF (SINGA.LT.0.0001) GO TO 102
        R = 1./SINGA
        JUMP = 105
C       PRINT 900,JUMP,JUMP2,JUMP3
        RETURN
 102    SINBE = SIN(BE)
        R = 1./SINBE
```

```
      JUMP = 106
C     PRINT 900,JUMP,JUMP3,JUMP2
      RETURN
C
C     Transformation entry point
C
 2    XX = X
      YY = Y
      ZZ = Z
C     PRINT 901,XX,YY,ZZ
C     PRINT 900,JUMP3
      IF (JUMP3.EQ.104) GO TO 104
 103  IF (ZZ.EQ.SPV) GO TO 109
 104  DENOM = (XX-EX)*COSAL + (YY-EY)*COSBE + (ZZ-EZ)*COSGA
      IF (DENOM.NE.0.0) GO TO 1111
      Q = 1.
      GO TO 50
 1111 Q = D/DENOM
 50   CONTINUE
C     PRINT 901,DENOM,Q
C     PRINT 900,JUMP
      IF (JUMP.EQ.106) GO TO 106
 105  XX = ((EX+Q*(XX-EX)-AX)*COSBE-(EY+Q*(YY-EY)-AY)*COSAL)*R
      YY = (EZ+Q*(ZZ-EZ)-AZ)*R
      GO TO 107
 106  XX = ((EZ+Q*(ZZ-EZ)-AZ)*COSAL-(EX+Q*(XX-EX)-AX)*COSGA)*R
      YY = (EY+Q*(YY-EY)-AY)*R
 107  IF (JUMP2.EQ.110) GO TO 110
 108  XX = AMIN1(U4,AMAX1(U1,U1+U3*(FACT*XX-UO)))
      YY = AMIN1(V4,AMAX1(V1,V1+V3*(FACT*YY-VO)))
      GO TO 110
 109  XX = NSPVAL
      YY = NSPVAL
      GO TO 110
 110  XT = XX
      YT = YY
C     PRINT 901,XT,YT
C     PRINT 900,JUMP2
      RETURN
 3    CONTINUE
      WRITE (3'NN)JUMP,JUMP2,JUMP3,EX,EY,EZ,COSAL,COSBE,COSGA,D,
     * AX,AY,AZ,R,UO,U1,U2,U3,U4,VO,V1,V2,V3,V4
C     PRINT 900,JUMP,JUMP2,JUMP3
C     PRINT 901,EX,EY,EZ,COSAL,COSBE,COSGA,D,AX,AY,AZ,R
C     PRINT 901,UO,U1,U2,U3,U4,VO,V1,V2,V3,V4
      RETURN
 4    CONTINUE
      READ (4'NN)JUMP,JUMP2,JUMP3,EX,EY,EZ,COSAL,COSBE,COSGA,D,
     * AX,AY,AZ,R,UO,U1,U2,U3,U4,VO,V1,V2,V3,V4
C     PRINT 900,JUMP,JUMP2,JUMP3
C     PRINT 901,EX,EY,EZ,COSAL,COSBE,COSGA,D,AX,AY,AZ,R
C     PRINT 901,UO,U1,U2,U3,U4,VO,V1,V2,V3,V4
      RETURN
      END
      FUNCTION ACOS(X)
```

```
      ACOS = 1./SQRT(1.+ATAN(X)*ATAN(X))
      END
      FUNCTION R(HO,HU)
C
C     This routine interpolates in the CV array.
C
      COMMON /CVAL/CV
      IF (HO-HU.EQ.0) GO TO 10
      R = (HO-CV)/(HO-HU)
C     PRINT 900,CV,HO,HU,R
 900  FORMAT (1H 4(2X,E14.7))
      RETURN
 10   CONTINUE
      IF (HO-CV.LT.0.0) R = 0.0
      IF (HO-CV.GE.0.0) R = 1.0
      RETURN
      END
```

```
      SUBROUTINE CTCELL (Z,MDZ,NX,NY,M,IO,JO)
C
C     CTCELL computes lines for constant Z (coutour lines) in one
C     cell of the array Z for the SRFACE package.
C     Z, NX, NY are the same as in SRFACE.
C             By the time ctcell is first called, M contains
C             the two-space plotter location of each Z point.
C             U(Z(I,J))=M(1,I,J), V(Z(I,J))=M(2,I,J)
C     IO,JO   The cell Z(I1,J1) to Z(I1+1,J1+1) is the one to
C             be contoured.
      DIMENSION Z(1089),M(2178)
      COMMON /SRFBLK/ LIMU(1024) ,LIML(1024) ,CL(41)    ,NCL    ,
     1             LL          ,FACT         ,IROT      ,NDRZ     ,
     2            NUPPER       ,NRSWT        ,BIGD      ,UMIN     ,
     3            UMAX    ,VMIN        ,VMAX      ,RZERO    ,
     4            NOFFP        ,NSPVAL       ,SPV       ,BIGEST
      COMMON /CVAL/ CV
      I1 = IO
      I1P1 = I1 + 1
      J1 = JO
      J1P1 = J1 + 1
      H1 = Z(I1+NX*(J1-1))
      H2 = Z(I1+NX*(J1P1-1))
      H3 = Z(I1P1+NX*(J1P1-1))
      H4 = Z(I1P1+NX*(J1-1))
C     PRINT 901,H1,H2,H3,H4
 901  FORMAT (1H 9E14.7)
      IF (NOFFP.NE.1) GO TO 101
      IF (H1.EQ.SPV.OR.H2.EQ.SPV.OR.H3.EQ.SPV.OR.H4.EQ.SPV) RETURN
 101  IF (AMIN1(H1,H2,H3,H4).GT.CL(NCL)) RETURN
C
C     For each contour level, decide which of the 16 basic
C     siturations exists, then interpolate in two-space to find
C     the end points of the contour line segment within this cell.
C
      DO 111 K=1,NCL
      CV = CL(K)
C     PRINT 901,CV
      K1 = (IFIX(SIGN(1.,H1-CV))+1)/2
      K2 = (IFIX(SIGN(1.,H2-CV))+1)/2
      K3 = (IFIX(SIGN(1.,H3-CV))+1)/2
      K4 = (IFIX(SIGN(1.,H4-CV))+1)/2
      JUMP = 1+K1+K2*2+K3*4+K4*8
C     PRINT 900,K1,K2,K3,K4,JUMP
      IFLG = 4
      IF (JUMP.EQ.1.OR.JUMP.EQ.16) GO TO 10
C     PRINT 904,JUMP,I1,I1P1,J1,J1P1
C     PRINT 905,K1,H1,K2,H2,K3,H3,K4,H4,CV
 904  FORMAT (1H 5I10)
 905  FORMAT (1H 4(I3,2X,F5.2),2X,F5.2)
 10   CONTINUE
      A1 = FLOAT(M(2*I1-1+2*NX*(J1-1)))
      A2 = FLOAT(M(2*I1+2*NX*(J1-1)))
      A3 = FLOAT(M(2*I1-1+2*NX*(J1P1-1)))
      A4 = FLOAT(M(2*I1+2*NX*(J1P1-1)))
```

```
      A5 = FLOAT(M(2*I1P1-1+2*NX*(J1-1)))
      A6 = FLOAT(M(2*I1P1+2*NX*(J1-1)))
      A7 = FLOAT(M(2*I1P1-1+2*NX*(J1P1-1)))
      B1 = FLOAT(M(2*I1-1+2*NX*(J1P1-1))-M(2*I1-1+2*NX*(J1-1)))
      B2 = FLOAT(M(2*I1+2*NX*(J1P1-1))-M(2*I1+2*NX*(J1-1)))
      A8 = FLOAT(M(2*I1P1+2*NX*(J1P1-1)))
      B31 = FLOAT(M(2*I1P1-1+2*NX*(J1P1-1))-M(2*I1-1+2*NX*(J1P1-1)))
      B41 = FLOAT(M(2*I1P1+2*NX*(J1P1-1))-M(2*I1+2*NX*(J1P1-1)))
      B11 = FLOAT(M(2*I1P1-1+2*NX*(J1-1))-M(2*I1-1+2*NX*(J1-1)))
      B21 = FLOAT(M(2*I1P1+2*NX*(J1-1))-M(2*I1+2*NX*(J1-1)))
      B3 = FLOAT(M(2*I1P1-1+2*NX*(J1-1))-M(2*I1P1-1+2*NX*(J1P1-1)))
      B4 = FLOAT(M(2*I1P1+2*NX*(J1-1))-M(2*I1P1+2*NX*(J1P1-1)))
      IF (JUMP.EQ.1) GO TO 112
      IF (JUMP.EQ.2) GO TO 103
      IF (JUMP.EQ.3) GO TO 105
      IF (JUMP.EQ.4) GO TO 106
      IF (JUMP.EQ.5) GO TO 107
      IF (JUMP.EQ.6) GO TO 102
      IF (JUMP.EQ.7) GO TO 108
      IF (JUMP.EQ.8) GO TO 109
      IF (JUMP.EQ.9) GO TO 109
      IF (JUMP.EQ.10) GO TO 108
      IF (JUMP.EQ.11) GO TO 104
      IF (JUMP.EQ.12) GO TO 107
      IF (JUMP.EQ.13) GO TO 106
      IF (JUMP.EQ.14) GO TO 105
      IF (JUMP.EQ.15) GO TO 103
      IF (JUMP.EQ.16) GO TO 111
102   IDUB = 1
103   RA = R(H1,H2)
      MUA = A1 + RA*B1
      MVA = A2 + RA*B2
      RB = R(H1,H4)
      MUB = A1 + RB*B11
      MVB = A2 + RB*B21
      GO TO 110
104   IDUB = -1
105   RA = R(H2,H1)
      MUA = A3 - RA*B1
      MVA = A4 - RA*B2
      RB = R(H2,H3)
      MUB = A3 + RB*B31
      MVB = A4 + RB*B41
      GO TO 110
106   RA = R(H2,H3)
      MUA = A3 + RA*B31
      MVA = A4 + RA*B41
      RB = R(H1,H4)
      MUB = A1 + RB*B11
      MVB = A2 + RB*B21
      GO TO 110
107   RA = R(H3,H2)
      MUA = A7 - RA*B31
      MVA = A8 - RA*B41
      RB = R(H3,H4)
```

```
      MUB = A7 + RB*B3
      MVB = A8 + RB*B4
      IDUB = 0
      GO TO 110
  108 RA = R(H2,H1)
      MUA = A3 - RA*B1
      MVA = A4 - RA*B2
      RB = R(H3,H4)
      MUB = A7 + RB*B3
      MVB = A8 + RB*B4
      GO TO 110
  109 RA = R(H4,H1)
      MUA = A5 - RA*B11
      MVA = A6 - RA*B21
      RB = R(H4,H3)
      MUB = A5 - RB*B3
      MVB = A6 - RB*B4
      IDUB = 0
  110 CONTINUE
C     PRINT 901,RA,RB
C     PRINT 900,MUA,MVA,MUB,MVB,IDUB,IFLG
      CALL DRAW(MUA,MVA,MUB,MVB,1)
      IFLG = 4
  900 FORMAT (1H 13I10)
      IF (IDUB) 109,111,107
  111 CONTINUE
  112 RETURN
      END
```

```
      SUBROUTINE DRAW(MX1,MY1,MX2,MY2,KFLAG)
C
C     This routine draws the visible part of the line connecting
C     (MX1,MY1) and (MX2,MY2).  The variable KFLAG is used to
C     specify which mode the subroutine uses:
C           1-Draw visible part of line
C           2-MARKs the visibility arrays
C           3-Both marks and draws
C
      LOGICAL*1 VIS1  ,VIS2
      COMMON /SRFBLK/ LIMU(1024) ,LIML(1024) ,CL(41)    ,NCL ,
     *                LL        ,FACT        ,IROT     ,NDRZ      ,
     *          NUPPER          ,NRSWT       ,BIGD     ,UMIN      ,
     *          UMAX     ,VMIN          ,VMAX     ,RZERO     ,
     *          NOFFP           ,NSPVAL      ,SPV      ,BIGEST
      DATA STEEP/5./
      ITFG = 1
      IF (KFLAG.LT.4) GO TO 55
      ITFG = KFLAG
      KFLAG = KFLAG - 3
 55   GO TO (1,2,3) KFLAG
C
C     DRAW
C
 1    IDRAW = 1
      IMARK = 0
C     PRINT 900,MX1,MY1,MX2,MY2,KFLAG,IDRAW,IMARK
 900  FORMAT(1HC 12I10)
      GO TO 101
C
C     MARK
C
 2    IDRAW = 0
      IMARK = 1
      GO TO 101
C
C     DRAW and MARK
C
 3    IDRAW = 1
      IMARK = 1
C
C     MARK line left to right.
C
 101  MMX1 = MX1
      MMY1 = MY1
      MMX2 = MX2
      MMY2 = MY2
      IF (MMX1.EQ.NSPVAL.OR.MMX2.EQ.NSPVAL) RETURN
      LOGICAL*1 BAD
      BAD = MX1.GT.1024
      BAD = MY1.GT.1024.OR.BAD
      BAD = MX2.GT.1024.OR.BAD
      BAD = MY2.GT.1024.OR.BAD
      BAD = MX1.LT.1 .OR.BAD
      BAD = MY1.LT.1 .OR.BAD
```

```
      BAD = MX2.LT.1 .OR.BAD
      BAD = MY2.LT.1 .OR.BAD
      IF(BAD) RETURN
      IF (MMX1.GT.MMX2) GO TO 102
      NX1 = MMX1
      NY1 = MMY1
      NX2 = MMX2
      NY2 = MMY2
      GO TO 103
 102  NX1 = MMX2
      NY1 = MMY2
      NX2 = MMX1
      NY2 = MMY1
 103  IF (NUPPER.LT.0) GO TO 119
C
C     Check upper visibility.
C
      VIS1 = NY1.GE.(LIMU(NX1)-1)
      VIS2 = NY2.GE.(LIMU(NX2)-1)
C     PRINT 900,NX1,NY1,NX2,NY2,VIS1,VIS2,LIMU(NX1),LIMU(NX2)
C
C     VIS1 and VIS2 TRUE means visible.
C
      IF (VIS1.AND.VIS2) GO TO 113
C
C     VIS1 or VIS2 false means invisible.
C
      IF (.NOT.(VIS1.OR.VIS2)) GO TO 119
C
C     Find change point.
C
      IF (NX1.EQ.NX2) GO TO 112
      DY = FLOAT(NY2-NY1)/FLOAT(NX2-NX1)
      NX1P1 = NX1 + 1
      FNY1 = NY1
      IF (VIS1) GO TO 107
      DO 104 K=NX1P1,NX2
      MX = K
      MY = FNY1 + FLOAT(K-NX1)*DY
      IF (MY.GT.LIMU(K)) GO TO 105
 104  CONTINUE
 105  IF (ABS(DY).GE.STEEP) GO TO 110
 106  NX1 = MX
      NY1 = MY
      GO TO 113
 107  DO 108 K=NX1P1,NX2
      MX = K
      MY = FNY1 + FLOAT(K-NX1)*DY
      IF (MY.LT.LIMU(K)) GO TO 109
 108  CONTINUE
 109  IF (ABS(DY).GE.STEEP) GO TO 111
      NX2 = MX
      NY2 = MY
      GO TO 113
 110  IF (LIMU(MX).EQ.0) GO TO 106
```

```
      NX1 = MX
      NY1 =LIMU(NX1)
      GO TO 113
 111  NX2 = MX
      NY2 = LIMU(NX2)
      GO TO 113
 112  IF (VIS1) NY2=MIN0(LIMU(NX1),LIMU(NX2))
      IF (VIS1) NY1=MIN0(LIMU(NX1),LIMU(NX2))
 113  IF (IDRAW.EQ.0) GO TO 116
C
C     Draw visible part of line.
C
      IF (IROT) 114,115,114
 114  CALL SEG(NY1,1023-NX1,NY2,1023-NX2,ITFG)
      GO TO 116
 115  CALL SEG(NX1,NY1,NX2,NY2,ITFG)
 116  IF (IMARK.EQ.0) GO TO 119
      IF (NX1.EQ.NX2) GO TO 118
      DY = FLOAT(NY2-NY1)/FLOAT(NX2-NX1)
      FNY1 = NY1
      DO 117 K=NX1,NX2
         LIMU(K) = FNY1 + FLOAT(K-NX1)*DY
 117  CONTINUE
      GO TO 119
 118  LIMU(NX1) = MAX0(NY1,NY2)
 119  IF (NUPPER) 120,120,138
C
C     Same idea as above, but for lower side.
C
 120  IF (MMX1.GT.MMX2) GO TO 121
      NX1 = MMX1
      NY1 = MMY1
      NX2 = MMX2
      NY2 = MMY2
      GO TO 122
 121  NX1 = MMX2
      NY1 = MMY2
      NX2 = MMX1
      NY2 = MMY1
 122  VIS1 = NY1.LE.(LIML(NX1)+1)
      VIS2 = NY2.LE.(LIML(NX2)+1)
      IF (VIS1.AND. VIS2) GO TO 132
      IF (.NOT.(VIS1.OR.VIS2)) GO TO 138
      IF (NX1.EQ.NX2) GO TO 131
      DY = FLOAT(NY2-NY1)/FLOAT(NX2-NX1)
      NX1P1 = NX1 + 1
      FNY1 = NY1
      IF (VIS1) GO TO 126
      DO 123 K=NX1P1,NX2
         MX = K
         MY = FNY1 + FLOAT(K-NX1)*DY
         IF (MY.LT.LIML(K)) GO TO 124
 123  CONTINUE
 124  IF (ABS(DY).GE.STEEP) GO TO 129
 125  NX1 = MX
```

```
      NY1 = MY
      GO TO 132
126   DO 127 K=NX1P1,NX2
            MX = K
            MY = FNY1 + FLOAT(K-NX1)*DY
            IF (MY.GT.LIML(K)) GO TO 128
127   CONTINUE
128   IF (ABS(DY).GE.STEEP) GO TO 130
      NX2 = MX
      NY2 = MY
      GO TO 132
129   IF (LIML(MX).EQ.1024) GO TO 125
      NX1 = MX
      NY1 = LIML(NX1)
      GO TO 132
130   NX2 = MX
      NY2 = LIML(NX2)
      GO TO 132
131   IF (VIS1) NY2 = MAX0(LIML(NX1),LIML(NX2))
      IF (VIS2) NY1 = MAX0(LIML(NX1),LIML(NX2))
132   IF (IDRAW.EQ.0) GO TO 135
      IF (IROT) 133,134,133
133   CALL SEG(NY1,1023-NX1,NY2,1023-NX2,ITFG)
      GO TO 135
134   CALL SEG(NX1,NY1,NX2,NY2,ITFG)
135   IF (IMARK.EQ.0) GO TO 138
      IF (NX1.EQ.NX2) GO TO 137
      DY = FLOAT(NY2-NY1)/FLOAT(NX2-NX1)
      FNY1 = NY1
      DO 136 K=NX1,NX2
            LIML(K) = FNY1 + FLOAT(K-NX1)*DY
136   CONTINUE
      RETURN
137   LIML(NX1) = MIN0(NY1,NY2)
138   RETURN
      END
```

```
      PROGRAM PLOTER
      LOGICAL*1 FIRST,YY,IY
      INTEGER X1GA(7),X2GA(7),Y1GA(7),Y2GA(7),XAA(17),XBA(17),
     * XCA(17),XDA(17)
      INTEGER CHEAD(48),IHEAD(10),ALP,ACM,ARP
      DIMENSION CVALQ(8),S(3),JHEAD(20)
      DATA YY/'Y'/
      DATA ALP/' ('/,ARP/') '/,ACM/' ,'/,FIRST/.TRUE./
      NC = 0
      NN = 0
      DLINE = 4.
      FMAG = .8
      TYPE *,' THE CURRENT PLOT MAGNIFICATION IS .8'
      TYPE *,' DO YOU WANT TO CHANGE PLOT MAGNIFICATION? (Y/N)'
      ACCEPT 9050,IY
 9050 FORMAT (A1)
      IF (IY.GT.98) IY = IY - 32
      IF (IY.NE.YY) GO TO 5
      TYPE *,' ENTER MAGNIFICATION FACTOR-DECIMAL'
      ACCEPT 906,FMAG
 906  FORMAT (F12.7)
 5    CONTINUE
      CALL PLOTS(0,0,0)
      CALL PLOT (0.0,0.75,-3)
      HT = 8.0
      NO = 1
      OPEN(UNIT=4,NAME='PLOT.DAT',TYPE='OLD',FORM='UNFORMATTED',
     * ACCESS='DIRECT',ASSOCIATEVARIABLE=NO,RECORDSIZE=10)
      DO 6 K=1,4
      READ (4'NO) HT,(CHEAD(I),I=1,13),NVAL
C     PRINT 900,NVAL
C     PRINT 901,HT
      CALL SYMBOL (0.0,HT,.1,CHEAD(1),0.0,24)
 6    CONTINUE
      READ (4'NO) X1G,X2G,Y1G,Y2G,ZCMAX
      READ (4'NO) S(1),S(2),S(3),NY,NX
      NY = NY - 1
      NX = NX - 1
 66   DO 7 I=1,NVAL
      READ (4'NO) HT,CVALQ
C     PRINT 900,NVAL
C     PRINT 901,HT
      CALL SYMBOL (0.0,HT,.1,CVALQ,0.0,12)
 7    CONTINUE
      READ (4'NO) XDIST,IHEAD
      TYPE *,' ENTER 40 CHARACTER PLOT TITLE'
      ACCEPT 9055,JHEAD
 9055 FORMAT (20A2)
C     PRINT 901,XDIST
      CALL SYMBOL (XDIST+1.,6.5,.1,JHEAD,0.0,40)
 10   CONTINUE
      AXO = AX
      AYO = AY
      READ (4'NO) AX,AY,IPC
      AX = AX*FMAG + 1.
```

```
      AY = AY*FMAG - .75
      IF (AXO.NE.0.0) GO TO 11
      XD = AX
      YD = AY
 11   YOMAX = YMAX
      YMAX = AMAX1(YOMAX,AY)
      IF (YOMAX.EQ.YMAX) GO TO 12
      XO = AX
 12   IF (NN.EQ.0.AND.FIRST) GO TO 18
      IF (ABS(AXO-AX).GT.DLINE) NN = NN + 1
      IF (NN.NE.0) GO TO 13
      XB = AX
      YB = AY
 13   IF (ABS(S(1)).LT.ABS(S(2))) GO TO 131
      IF (NN.NE.NX) GO TO 18
      GO TO 132
 131  IF (NN.NE.NY) GO TO 18
 132  IF (NC.NE.0) GO TO 14
      XC = AX
      YC = AY
      NC = NC + 1
 14   CONTINUE
      XA = AXO
      YA = AYO
 18   FIRST = .FALSE.
 900  FORMAT (1H 13010)
C     PRINT 903,AX,AY,IPC,NO,NC,AXO,AYO
 903  FORMAT (1H 2F12.4,3I7,2F12.4)
 901  FORMAT (1H 9F12.4)
      IF (IPC.EQ.0) GO TO 20
      IF (AX.EQ.-9999.) GO TO 20
      CALL PLOT (AX,AY,IPC)
      GO TO 10
 902  FORMAT (F6.3)
 20   CONTINUE
      IF (XA.LT.XC) GO TO 21
      XS = XC
      XC = XA
      XA = XS
      YS = YC
      YC = YA
      YA = YS
 21   CONTINUE
      IF (XD.GT.XB) GO TO 22
      XS = XB
      XB = XD
      XD = XS
      YS = YB
      YB = YD
      YD = YS
 22   CONTINUE
C     TYPE 901,XA,YA,XB,YB,XC,YC,XD,YD,X1G,X2G,Y1G,Y2G
      ENCODE(6,902,X1GA)X1G
      ENCODE(6,902,X2GA)X2G
      ENCODE(6,902,Y1GA)Y1G
```

```
      ENCODE(6,902,Y2GA)Y2G
C     TYPE 905,X1GA
C     TYPE 905,X2GA
C     TYPE 905,Y1GA
C     TYPE 905,Y2GA
 905  FORMAT (1H 28A2)
      IF (ABS(S(1)).LT.ABS(S(2))) GO TO 200
      IF (S(1).LT.0.0) GO TO 100
      XAA(1) = ALP
      DO 25 I=1,3
 25   XAA(I+1) = X1GA(I)
      XAA(5) = ACM
      DO 30 I=1,3
 30   XAA(I+5) = Y1GA(I)
      XAA(9) = ARP
C     TYPE 905,XAA
      XBA(1) = ALP
      DO 35 I=1,3
 35   XBA(I+1) = X2GA(I)
      XBA(5) = ACM
      DO 40 I=1,3
 40   XBA(I+5) = Y1GA(I)
      XBA(9) = ARP
C     TYPE 905,XBA
      XCA(1) = ALP
      DO 45 I=1,3
 45   XCA(I+1) = X1GA(I)
      XCA(5) = ACM
      DO 50 I=1,3
 50   XCA(I+5) = Y2GA(I)
      XCA(9) = ARP
C     TYPE 905,XCA
      XDA(1) = ALP
      DO 55 I=1,3
 55   XDA(I+1) = X2GA(I)
      XDA(5) = ACM
      DO 60 I=1,3
 60   XDA(I+5) = Y2GA(I)
      XDA(9) = ARP
      GO TO 500
 100  XAA(1) = ALP
      DO 125 I=1,3
 125  XAA(I+1) = X2GA(I)
      XAA(5) = ACM
      DO 130 I=1,3
 130  XAA(I+5) = Y2GA(I)
      XAA(9) = ARP
      XBA(1) = ALP
      DO 135 I=1,3
 135  XBA(I+1) = X1GA(I)
      XBA(5) = ACM
      DO 140 I=1,3
 140  XBA(I+5) = Y2GA(I)
      XBA(9) = ARP
      XCA(1) = ALP
```

```
      DO 145 I=1,3
145   XCA(I+1) = X2GA(I)
      XCA(5) = ACM
      DO 150 I=1,3
150   XCA(I+5) = Y1GA(I)
      XCA(9) = ARP
      XDA(1) = ALP
      DO 155 I=1,3
155   XDA(I+1) = X1GA(I)
      XDA(5) = ACM
      DO 160 I=1,3
160   XDA(I+5) = Y1GA(I)
      XDA(9) = ARP
      GO TO 500
200   IF (S(2).LT.0.0) GO TO 300
      XAA(1) = ALP
      DO 225 I=1,3
225   XAA(I+1) = X2GA(I)
      XAA(5) = ACM
      DO 230 I=1,3
230   XAA(I+5) = Y1GA(I)
      XAA(9) = ARP
      XBA(1) = ALP
      DO 235 I=1,3
235   XBA(I+1) = X2GA(I)
      XBA(5) = ACM
      DO 240 I=1,3
240   XBA(I+5) = Y2GA(I)
      XBA(9) = ARP
      XCA(1) = ALP
      DO 245    I=1,3
245   XCA(I+1) = X1GA(I)
      XCA(5) = ACM
      DO 250 I=1,3
250   XCA(I+5) = Y1GA(I)
      XCA(9) = ARP
      XDA(1) = ALP
      DO 255 I=1,3
255   XDA(I+1) = X1GA(I)
      XDA(5) = ACM
      DO 260 I=1,3
260   XDA(I+5) = Y2GA(I)
      XDA(9) = ARP
      GO TO 500
300   XAA(1) = ALP
      DO 325 I=1,3
325   XAA(I+1) = X1GA(I)
      XAA(5) = ACM
      DO 330 I=1,3
330   XAA(I+5) = Y2GA(I)
      XAA(9) = ARP
      XBA(1) = ALP
      DO 335 I=1,3
335   XBA(I+1) = X1GA(I)
      XBA(5) = ACM
```

```
      DO 340 I=1,3
 340  XBA(I+5) = Y1GA(I)
      XBA(9) = ARP
      XCA(1) = ALP
      DO 345 I=1,3
 345  XCA(I+1) = X2GA(I)
      XCA(5) = ACM
      DO 350 I=1,3
 350  XCA(I+5) = Y2GA(I)
      XCA(9) = ARP
      XDA(1) = ALP
      DO 355 I=1,3
 355  XDA(I+1) = X2GA(I)
      XDA(5) = ACM
      DO 360 I=1,3
 360  XDA(I+5) = Y1GA(I)
      XDA(9) = ARP
C     TYPE 905,XDA
 500  CALL SYMBOL (XB-.5,YB-.25,.075,XBA,0.0,17)
      CALL SYMBOL (XA-.75,YA+.25,.075,XAA,0.0,17)
      CALL SYMBOL (XC-.75,YC+.25,.075,XCA,0.0,17)
      CALL SYMBOL (XD-.75,YD-.25,.075,XDA,0.0,17)
      CALL NUMBER (XO,YMAX+.1,.1,ZCMAX,0.0,1)
      CALL PLOT(0.,0.,+999)
      STOP
      END
```

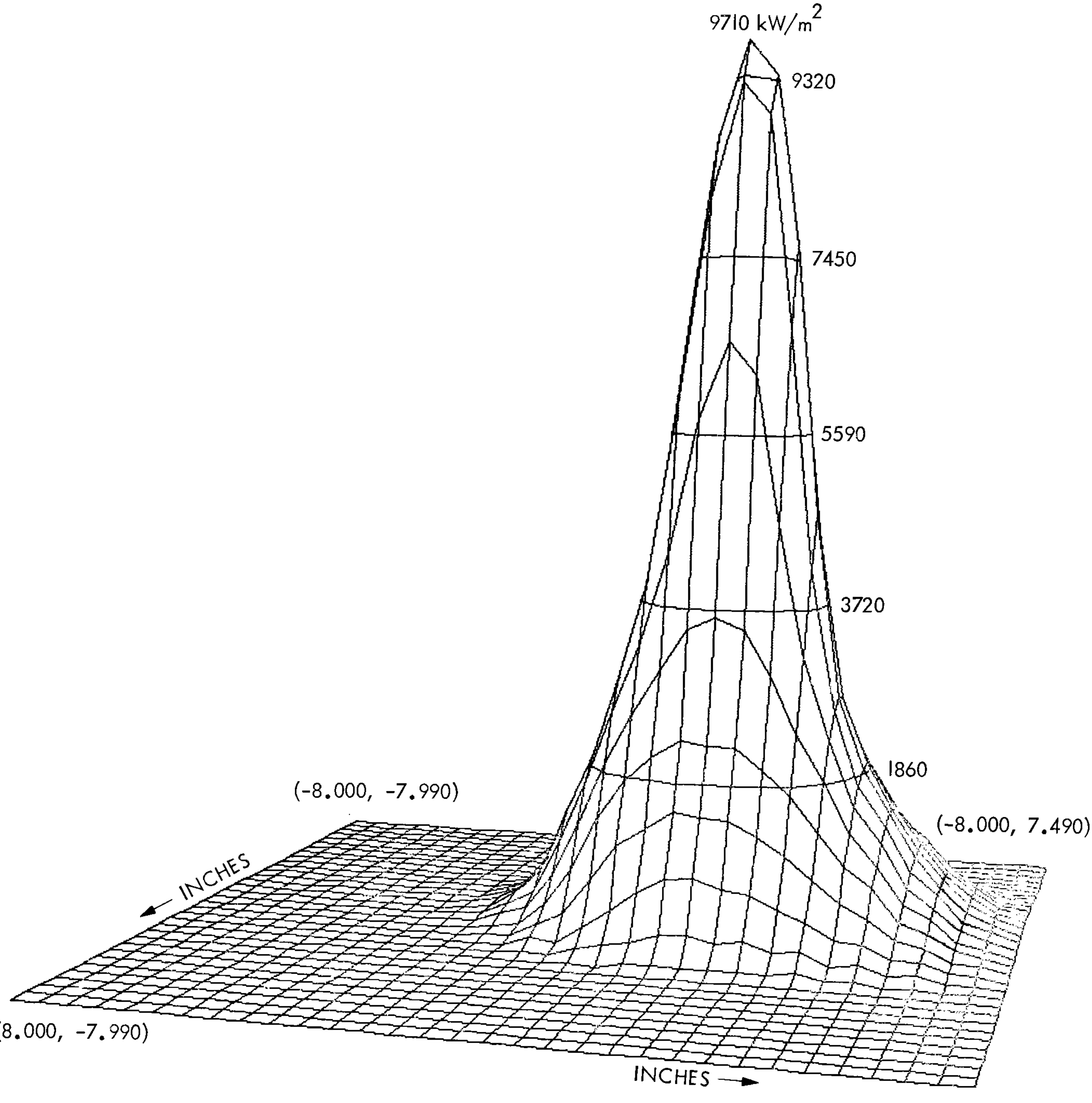

Figure B-1. A Typical Flux Map Along a Concentrator Focal Plane

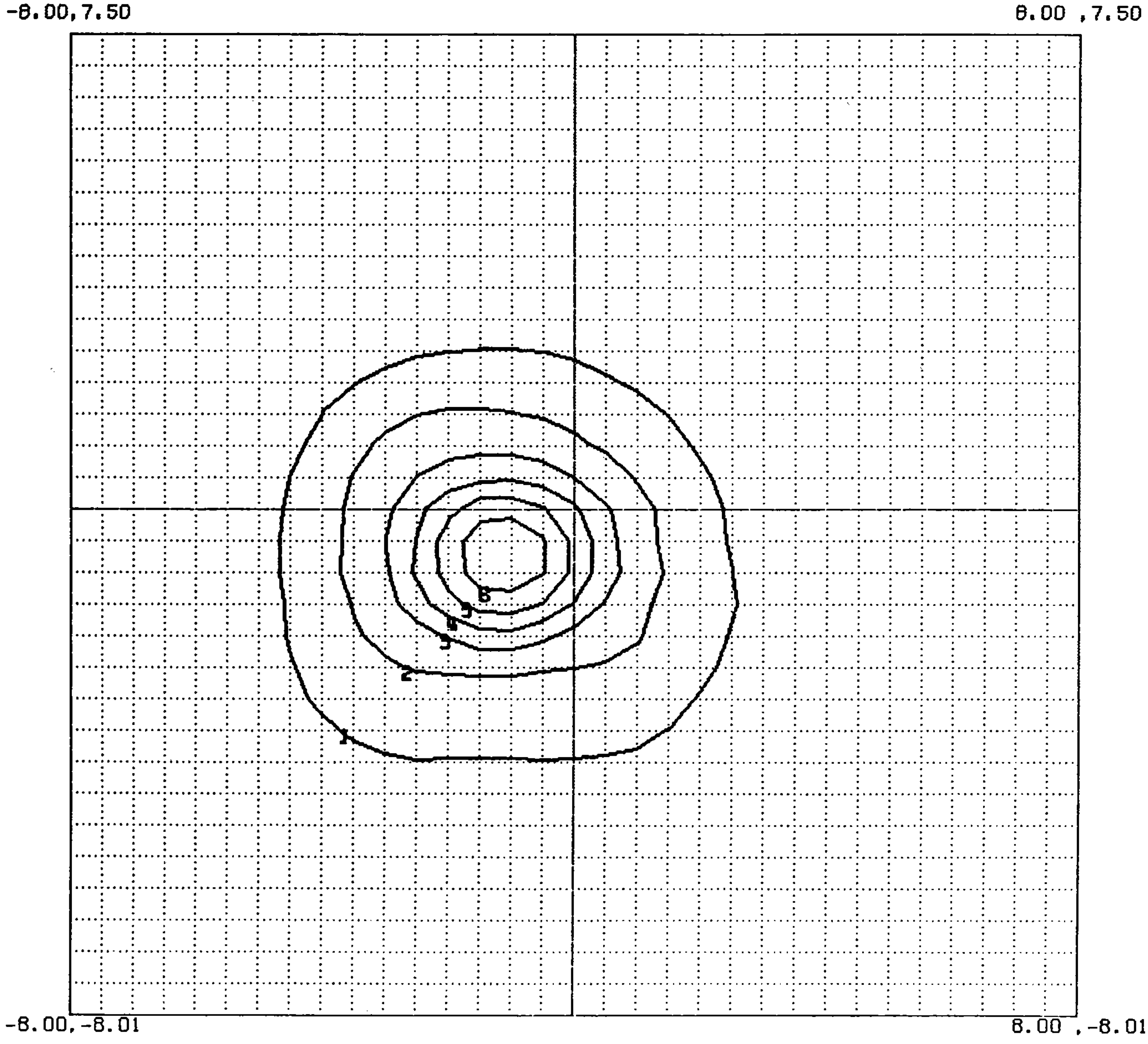

Figure B-2. A Typical Contour Map at a Concentrator Focal Plane (Contour software not listed in this report, but available from SNLA.)