LFASE DATA TRANSCRIPTION SOFTWARE:

CGG2ROSE2

Version 1.1

Program Report

by

W. S. Little

Information Processing and Communications Laboratory

N. R. Galbraith

Arctic Seismics and Acoustics Laboratory

S. T. Bolmer

Geology and Geophysics Department

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts

May, 1990

ABSTRACT

Seismic data obtained using borehole seismometers during the LFASE experiment (Stephen et al, 1989; Koelsch et al, 1990) are retrieved from the seafloor instrument package and converted to a data format more suitable for analysis with existing computer systems. This report describes the computer program CGG2ROSE2 for converting LFASE optical disk files recorded in CGG format by the data acquisition computer into standard ROSE format data files on a VAX/VMS computer. CGG2ROSE2 provides features for rearranging LFASE data into smaller segments, making corrections to timing information, and replacing file header information.

Contents

1	INTRODUCTION	1
1.1	Report Overview	1
1.2	CGG Data Acquisition and Preprocessing	2
2	PROGRAM OVERVIEW	2
2.1	File Utilization	2
2.2	Data Organization	4
2.3	Header Modification	4
2.4	Data Conversion	4
2.5	BCU Electronics and Scan Counter Verification	5
2.6	Alternate ROSE Headers	6
2.7	Program Modes	6
3	PROGRAM LOGIC	8
3.1	Library Subprogram Utilization	10
3.2	Program Structure	10
4	INPUT/OUTPUT	10
4.1	Input Files	10
4.2	Output Files	12
4.3	Scratch File Usage	13
4.4	File Naming Conventions	13
4.5	ROSE File Names	13
4.6	LOG File Names	14
4.7	File Directory Requirements	15
4.8	Record Formats	15
5	SUBPROGRAM DESCRIPTIONS	16

5.1	OPDOPEN (NRG)	16
5.2	OPDRDHED (NRG)	16
5.3	OPDRDTR (NRG)	17
5.4	OPDRDAT (NRG)	17
5.5	GETPAR (NRG)	17
5.6	CGGVAX (NRG)	18
5.7	GETSTAT (NRG)	18
5.8	BYTESWAP (NRG)	19
5.9	SETIME (NRG)	19
5.10	OUTOPEN (NRG)	20
5.11	WRTOUT (NRG)	20
5.12	WRAPUP (NRG)	21
5.13	WRAPUP2 (WSL)	21
5.14	HEADER (NRG)	22
5.15	HEADER2 (WSL)	23
5.16	ROSETIME (WSL)	23
5.17	CALCDATE, CALCTIME, CALCYMD, ADDMIL, ACCUM (WSL)	25
6	PROGRAM OPERATION	25
6.1	Compilation	26
6.2	Linking	26
6.3	Execution	27
7	RESTRICTIONS	27
8	RECOMMENDED IMPROVEMENTS	29

APPENDICES

Appendix 1: Sample Runs	33
Appendix 2: Parameter File Formats (and READ statements)	44
Appendix 3: CGG File Structure	45
Appendix 4: WHOI ROSE File Format	50
Appendix 5: Related Programs	56
Appendix 6: CGG2ROSE2 Program History	58

List of Figures

1	CGG2ROSE2 Version 1.1 Program Overview	3
2	CGG2ROSE2 Operating Modes	7
3	CGG2ROSE2 Program Logic	9
4	CGG2ROSE2 Program Structure	11

1 INTRODUCTION

CGG2ROSE2 is a program which converts CGG formatted seismic data files on optical disk to ROSE formatted files on magnetic disk. The program was developed for processing data acquired during the LFASE borehole seismic experiment.

This report describes Version 1.1 of CGG2ROSE2.

1.1 Report Overview

Section 1 summarizes this Program Report and briefly describes CGG data acquisition and preprocessing.

Section 2 is a description of the operating modes of CGG2ROSE2.

Section 3 describes the program logic.

Section 4 discusses CGG2ROSE2 input/output file usage.

Section 5 gives detailed information about the subprograms used by CGG2ROSE2.

Section 6 provides systems and applications programming information such as command files for compiling, linking and running the program.

Section 7 lists operating restrictions.

Section 8 includes suggested future program improvements.

The report concludes with Appendices 1 through 6 which contain program examples, parameter file record formats, the CGG file format, the ROSE file format, a summary of related programs, and some historical notes regarding CGG2ROSE2 program development.

“LFASE” is an acronym for Low Frequency Acoustic Seismic Experiment. “ROSE” stands for Riveira Ocean Seismic Experiment, an earlier scientific program during which this seismic data format was devised (Latraille, 1983; Latraille and Dorman, 1983). “CGG” means Compagnie Générale de Géophysique, manufacturer of LFASE borehole sensors, digitizer and telemetry hardware. The CGG system is described in reference CGG (1988a).

1.2 CGG Data Acquisition and Preprocessing

The CGG files are created using a Mountain Optech disk drive connected to the MS-DOS based computer system that is part of the seismic data acquisition system. After completion of the data acquisition phase of the LFASE experiment, the optical disk cartridges are removed from the disk drives in the instrument package and inserted into the disk drive on the VAX/VMS computer where they are converted to ROSE format using CGG2ROSE2.

Before using CGG2ROSE2, preprocessing may be performed on a separate MS-DOS based computer having a compatible optical disk drive in order to exercise greater quality control, improve data organization, generate preliminary time series plots, and create backup copies of raw data files.

2 PROGRAM OVERVIEW

Figure 1 gives an overview of the CGG2ROSE2 program. It shows the input and output files used by CGG2ROSE2. The program may be run interactively using the computer keyboard/display. However, for most LFASE data processing, the program is likely to be run as a batch processing job, using VAX/VMS command (COM) files in place of the on-line commands.

2.1 File Utilization

The program reads a series of CGG input data files, together with several input parameter files specifying which CGG files to process, which ROSE files to create, and what header information to associate with each output file. In addition to creating ROSE files, the program generates several listing files, as well as a file containing duplicate copies of all of the ROSE header records for use later in editing header information.

Figure 1: CGG2ROSE2 Version 1.1 Program Overview

2.2 Data Organization

Each input optical disk contains a disk directory and a number of individual CGG data files. Each data file contains CGG header information, blocked data scans each containing binary data values from twelve instrument channels, and trailer information. The program CGG2ROSE2 converts these files to standard ROSE files, each composed of ROSE header records followed by blocked binary data records. The output files are in "WHOI Standard ROSE format", in which the data are expressed as VAX 32-bit integers.

In addition to performing data conversion, header reformatting, and dividing individual files into shorter time series, the program "demultiplexes" the twelve individual data channels so that each ROSE file contains all channel 1 data followed by all channel 2 data, and so forth, as required by the ROSE format specification. Appendices 3 and 4 include detailed CGG and ROSE file format descriptions.

2.3 Header Modification

Header information from the CGG input files is reformatted by the program to conform to ROSE standards. ROSE file header information may be modified by the user in a number of ways. In program MODE 0 (see below) individual header values may be entered by using the "Primary Parameter File". In other program MODEs, alternate header data may be introduced using the "ROSE Date/Time Parameter File". In this manner, the start-of-data date/time, the event date/time, and shot-specific information such as the shot depth, range, and bearing may be introduced into the ROSE headers.

2.4 Data Conversion

Each individual seismic data value is converted from a CGG 16-bit number (2's complement mantissa plus scale factor) to a normalized 32-bit VAX integer.

For a 16-bit CGG data value expressed as

g g g g s m m m m m m m m m m m m

the corresponding 32-bit VAX data value is calculated using the formula

$$V = M * 2048 / (2^{**}G)$$

where

M = s m m m m m m m m m m m m is 12-bit 2's complement integer ranging from -2048 to +2047

and

G = g g g g is a 4-bit integer ranging from 0 to 11 (with values 12, 13, 14, and 15 not used).

Conversion is actually accomplished using lookup tables to speed program execution.

2.5 BCU Electronics and Scan Counter Verification

The Bottom Control Unit (BCU) electronics were designed by WHOI to provide an interface between the borehole array (originally designed to function on a seven conductor logging cable) and the shipboard control units. These are connected by high and low speed telemetry channels (over .068 inch coaxial cable) provided by Scripps Institution of Oceanography's Marine Physics Laboratory. One function of the BCU is to reduce the sample rate from the CGG digitizer from 500 samples/second to 125 samples/second. The engineering data on the auxiliary channels is also handled correctly by the BCU. The BCU also adds scan counter information to the CGG data stream so that telemetry and recording fidelity could be checked.

The CGG input data are checked by CGG2ROSE2 to insure that the scan counter value increments correctly from scan to scan. A scan is a group of 32 bytes comprising one 2-byte sample for each of twelve channels, plus sync bytes plus scan counter, plus other engineering diagnostic bytes. One scan is output every .008 seconds from the BCU.

2.6 Alternate ROSE Headers

CGG2ROSE2 writes an alternate ROSE header file containing copies of the ROSE headers that have been generated for the actual ROSE output files. This file is available for subsequent editing, should the original ROSE headers need to be replaced.

2.7 Program Modes

To facilitate piecewise ROSE file generation, CGG2ROSE2 Version 1.1 has five operating modes, shown schematically in Figure 2. These modes are:

MODE 0: Single ROSE file mode.

The program converts a series of CGG files to ROSE files. Each CGG file produces one full length ROSE file, that is a ROSE file having the same number of samples as the corresponding CGG file. (This was the only mode available in the earlier program version CGG2ROSE1.)

MODE 1: Multiple ROSE file mode.

The program converts a series of CGG files to ROSE files. Each CGG file may be divided into a number of time segments, each of which is converted to a separate ROSE file. Therefore, the program typically converts a small number of large CGG files into a large number of smaller ROSE files. ROSE output files may be optionally saved or skipped (omitted); this allows the user to create ROSE files containing only data of interest while ignoring the remaining data.

MODE 2: Multiple ROSE file mode (alternate).

The program operates identically to MODE 1, except that the selections of ROSE output files to be saved and those to be skipped, are reversed. This feature is useful for creating ROSE files which essentially fill in gaps between the ROSE files created using MODE 1.

Figure 2: CGG2ROSE2 Operating Modes

The selection of files to be saved or skipped is governed by "skip/save" codes which are specified by the user. Those ROSE files skipped during MODE 1 processing can later be generated using MODE 2; those files created by MODE 1 will be skipped when using MODE 2.

The use of skip/save codes facilitates piecewise processing of CGG data files; they are useful for deleting extraneous leading or trailing data, selecting small data windows for special processing, and so forth.

MODE 3: Multiple ROSE file mode with output suppressed.

The program operates identically to MODEs 1 and 2, except that no output files are produced. This is useful for verifying program input parameters such as the starting date/time values which have been specified for each ROSE output file.

MODE 4: Preview CGG start times and event times.

The program opens each specified CGG input file, extracts and prints the start-of-data date/time and the event date/time. These data are needed by the user in preparing the various program parameter files for subsequent MODE 1, 2, or 3 processing.

3 PROGRAM LOGIC

The logical operation of CGG2ROSE2 is presented in Figure 3. Parameters are first read from the various parameter files. A table is then constructed of starting and ending sample counts for each ROSE file to be constructed from a single CGG file. The CGG file is read and copied to several scratch files, one for each data channel. The scratch files are then copied to the series of ROSE files, one segment of each channel at a time. The process is repeated for each CGG file.

Figure 3: CGG2ROSE2 Program Logic

3.1 Library Subprogram Utilization

The program makes use of Mountain Optech VAX/VMS software library routines OOPEN, OREAD, OCLOSE. It also uses routines NAME, OPENOUT, and TIMEX from the WHOI G&G Department [SECURE]SLIB library, as well as DATE and TIME from the VAX/VMS library.

3.2 Program Structure

Figure 4 is a structure diagram of CGG2ROSE2. It shows the relationships among the main program and the various subprograms. In general, the subprograms in the diagram are arranged from left to right in order of first appearance. The arrows point from the calling module to the called module.

4 INPUT/OUTPUT

CGG2ROSE2 utilizes four kinds of input files, six kinds of output files, twelve intermediate scratch files, and the operator's keyboard/display.

4.1 Input Files

1. One or more CGG input data files on Mountain Optech optical disk.
2. A Primary Parameter File named CGG2ROSE2.PAR which contains:
  - (a) the names of the first and last optical disk files to be processed,
  - (b) the default event number for the first ROSE output file,
  - (c) the VAX/VMS disk and directory name for the ROSE output files,
  - (d) the instrument receiver number to use in writing ROSE output files,
  - (e) the VAX/VMS file extent for the ROSE output files,
  - (f) selected ROSE header data to be altered, and
  - (g) the code which determines the program MODE;

The use of item f for modifying ROSE header data is recommended only for MODE 0. For MODEs 1, 2, 3, or 4 header data should be altered by utilizing the appropriate fields of the ROSE Date/Time Parameter File (see below).

3. A ROSE Date/Time Parameter File containing date and time values, together with user specified event numbers and skip/save flags, which tell the program which segments of the series of data points to place in each ROSE file. This file optionally contains alternative start-of-data and event date/times to be written into the ROSE headers. This file also contains the depth, range, and bearing of the instrument, the shot depth, and other information to be added to the ROSE header (shot weight, bubble pulse time, flight time, latitude, longitude, and type of shot).
4. A CGG Filename Parameter File containing the names of the CGG input files to be processed.

4.2 Output Files

1. One or more (groups of) ROSE output files.
2. An alternate ROSE header file.
3. A LOG file containing information about the particular program execution.
4. A file listing the ROSE files created by the program from each CGG input file, along with the corresponding first sample times and block counts.
5. A file listing CGG files and their first sample and event times;
6. A file containing CGG data scan counter error messages.

4.3 Scratch File Usage

Twelve scratch files are written during the input phase of the program. The data from the CGG input file being processed are demultiplexed into twelve separate channels and written to each of the scratch files. The program then reads the scratch files and writes the data to a series of ROSE output files, each preceded by a ROSE header. For each ROSE file, the data are written one channel at a time. This process is repeated for each CGG input file.

4.4 File Naming Conventions

CGG2ROSE2 utilizes certain file naming conventions designed to avoid confusion when processing large numbers of nearly identical files in a production environment. The CGG input file names are determined by the data in the CGG Filename Parameter File. The Primary Parameter File name is hard-coded as CGG2ROSE2.PAR and is read using FORTRAN logical unit number 20. The other two parameter file names are determined by the user by assigning FORTRAN logical unit numbers 55 and 88 to the ROSE Date/Time Parameter File and the CGG Filename Parameter File respectively.

4.5 ROSE File Names

CGG2ROSE2 automatically establishes VAX/VMS file names for the ROSE output files, the LOG output file, and the alternate ROSE header output file.

The ROSE output files are named:

RaaEbbbb.ccc,

where

aa is the receiver number,

bbbb is the event number, and

ccc is the file extent specified in the Primary Parameter File.

The ROSE output files are written to VAX/VMS disk and directory:

dddd:[eeee.ff], or dddd:[eeee.gg]

where   dddd, eeee, ff   are the output (disk) device name, the experiment name, and the deployment number respectively, as specified in the Primary Parameter File.

ff should be an even number.

gg is set by the program to equal ff plus 1.

4.6 LOG File Names

The LOG file name and the alternate Rose header file names are:

RggEhhiij.LOG,
RggEhhiij.HDR,

where   gg, hh, ii, and j   are taken from the first CGG input filename used. The filename is itself composed of date and time data representing the approximate time the file was recorded by the data acquisition system.

gg is the day field from the CGG filename,
hh is the hours field,
ii is the minute fields, and
j is the first digit of the seconds field.

For example, a file recorded 01/19/89 at 20:55:26.321 will have a CGG file name assignment of 01192055.269 (the 9 in the .269 is from the year 89). The .LOG and .HDR files will be named R19E20553.LOG and R19E20553.HDR.

4.7 File Directory Requirements

CGG2ROSE2 requires that certain VAX/VMS file subdirectories be created prior to program execution, and that certain subdirectory naming conventions be observed. Specifically, two subdirectories must be created using the VMS CREATE DIRECTORY command. The top level of each subdirectory must be the experiment name, for example LFAS; the second level subdirectories must be two sequential numbers beginning with an even number corresponding to the deployment number, for example, 66 and 67. These values should correspond to those appearing in the Primary Parameter File. Thus, the two subdirectories [LFAS.66] and [LFAS.67] should be created before running CGG2ROSE2. Steps should be taken to insure that these subdirectories have the proper access privilege.

The deployment number should be an even number because CGG2ROSE2 uses two output subdirectories, one for ROSE files derived from noise data and the other for ROSE files derived from shot data. CGG2ROSE2 puts shot files in the subdirectory corresponding to the deployment number. It puts noise files in the subdirectory corresponding to the deployment number plus one.

4.8 Record Formats

The actual record formats corresponding to the input parameter files required frequent modification during the course of the LFASE experiment in order to conform to data formats provided by other investigators or generated automatically by other programs. The general format of each parameter file may be seen by referring to the sample program run in Appendix 1. The corresponding FORTRAN FORMAT statements are listed in Appendix 2.

As this report goes to press, further changes to the input parameter files are being made or anticipated. In the last analysis, the Fortran FORMAT statements in the latest version of the source code should be consulted when setting up parameter files, particularly for long computer runs.

5 SUBPROGRAM DESCRIPTIONS

The portions of the program which were contributed by N. Galbraith are marked "(NRG)". Those portions by W. Little are marked "(WSL)".

5.1 OPDOPEN (NRG)

Integer function OPDOPEN is called by the main program to open a single data file on optical disk. It calls the Mountain Optech function OOPEN.

There are no input arguments.

The return value is:

0 for success, -999 for error

Other values are passed in common files comname.for and comio.for.

5.2 OPDRDHED (NRG)

Integer function OPDRDHED is called by the main program to read a CGG file header from optical disk using Mountain Optech's applications library routine OREAD. The 2 sync bytes of the CGG header are read in a call to function GETSTAT.

After reading the CGG header, OPDRDHED calls subroutine setime to fill the time fields in the output ROSE header.

The input argument is:

integer isync, a flag which, if set to 1, denotes that the 2 sync bytes have been read already.

The return value is:

0 for end of file, -1 for error, 1 for success.

Other values are passed in common files comrose.for and comio.for.

5.3 OPDRDTR (NRG)

Integer function OPDRDTR reads a CGG file trailer from optical disk using Mountain Optech's applications library routine OREAD. This function is identical to OPDRDHED except that it is used only to obtain the scan count. It does not call SETIME to fill the ROSE time fields.

5.4 OPDRDAT (NRG)

Integer function OPDRDAT is called by the main program to read data from optical disk using Mountain Optech's applications library routine OREAD. It calls function GETSTAT to read the 2 sync bytes at the start of each data scan. It reads up to 2048 32-byte records of CGG data, and checks the expected 2 sync bytes at the start of each scan. It extracts 12 channel data words from input records by calling function BYTESWAP to create integers from the byte pairs in fields 3-8 and 11-16.

The input argument is:

integer isync equals 0 if sync bytes are to be read in this routine.

The output argument is:

integer isync returned as -1 for unexpected header
0 for data sync found as expected

The return value is:

0 for end of file, -1 for error,
scans read if successful

Other values are passed in common files comrose.for and comio.for.

5.5 GETPAR (NRG)

This function is called by the main program to read two parameter files, (1) the Primary Parameter File whose name is hard coded as CGG2ROSE2.PAR, and (2) the CGG filename file.

There are no input arguments.

The return value is:

-999 for error

0 , 1, 2, 3, or 4 for success, where the value returned is the program MODE.

5.6 CGGVAX (NRG)

This subroutine converts data values from CGG format to WHOI integer ROSE integer*4 format using the formula:

$$v = m * 2^{(11 - g)}$$

where m is a 12-bit 2's complement integer in the range [-2048,2047]

g is a power-of-2 exponent in the range [0,11]

v is a VAX/VMS 32-bit 2's complement integer in the range [-4194304,4192256]

The input arguments are:

iicgg = integer*2 array of seismic data values to be converted,
inum = length of iicgg and ival
= number of values to be converted.

The output arguments are:

ival = integer*4 array of converted seismic data values.

5.7 GETSTAT (NRG)

Integer function GETSTAT searches for 2 bytes of status (or sync bytes) using Mountain Optech's function OREAD. It reads and discards 1 byte at a time, up to 9999 bytes, until it has matched the byte pair that signal the beginning of a data scan, header, or trailer record.

The input argument is:

byte bstream(512) location of start of byte array of data

The return value is:

2 for data sync, 1 for header sync,
0 for end of file, -1 for error

Other values are passed in common files comio.for.

5.8 BYTESWAP (NRG)

Integer function BYTESWAP swaps high to low byte of a 2 byte array, putting the result in the integer return value. This is called by the function OPDRDAT to convert CGG data notation to VAX integer*2 notation.

The input argument is:

byte inbytes(2) the bytes to be swapped.

The return value is:

An integer*2 field containing the swapped bytes.

5.9 SETIME (NRG)

Subroutine SETIME is called by OPDRDHED to fill the time fields in the ROSE header. The input time fields are the W.R.C. clock, the shot clock, and the DOS clock. The output time fields are the data start time and the event time. For each output time field, the input time fields are searched in order of preference. Ideally, the W.R.C. clock is used to fill the data start time field, and the shot clock is used for the output event time. If for some reason an input time field is missing, another choice is made to fill the output time field.

nst/net	clock	preference for	
		start time	event time
3 =	w.r.c. clock	1	2
4 =	shot clock	2	1
2 =	dos clock	3	3
-1 =	no clock available		

The input argument is:

integer*2 hstruct(256) contains the CGG file header

The output argument is:

integer nst indicates which clock was used to fill the start time field

integer net indicates which clock was used to fill the event time field

Other values are passed in common files comrose.for and comio.for.

The W.R.C. clock uses the milliseconds field in a non-standard way.
The calling program CGG2ROSE2 sets it to zero.

5.10 OUTOPEN (NRG)

Integer function OUTOPEN is called by the main program to open all the output data files. It creates 12 scratch files for the channel data. It also calls subroutine NAME to name the ROSE output file and calls OPENOUT to open that file, both of which are in the WHOI G&G Department [SECURE] subroutine library.

There are no input arguments.

The return value is:

0 for success, -1 for error

All other values are passed through common, using files comio.for, com-name.for, and comrose.for.

5.11 WRTOUT (NRG)

Integer function WRTOUT is called by the main program in the data block loop. It writes 1 data block of 2048 samples to each of the 12 channel files. For the last block in an output file, the block length may be less.

The input argument is:

integer nobs is the number of scans to write

The return value is:

number of scans written, or -1 for error

Other values are passed in common files comrose.for, comio.for.

5.12 WRAPUP (NRG)

Subroutine WRAPUP is called by the main program after the last output scan has been written. It calls subroutine HEADER to create and write the ROSE file header. It then 'rewinds' the 12 channel files by opening and closing them. It reads the data in blocks of 2048 scans, calls subroutine CGGVAX to convert the scans to VAX/ROSE notation, and writes them to the ROSE file. The scratch files are then deleted, and the ROSE file is closed.

The input arguments are:

integer nblks is number of blocks

integer nout is the number of values in last record

The output argument is:

integer istat, 0 = success, negative = error.

Other values are passed in common files comrose.for and comio.for.

5.13 WRAPUP2 (WSL)

The subroutine WRAPUP2 is called by the main program CGG2ROSE2 to handle the final stage of conversion of seismic data files from CGG to ROSE format. It is used by all MODEs of operation of CGG2ROSE2 except MODE 0, which instead uses the subroutine WRAPUP.

The input arguments are:

nblks = the number of blocks in each input scratch file,

nout = the number of samples in the last block in each scratch file,
usually a short block.

The output argument is:

istat = 0 for normal return,

istat = -999 for file open, read, or write errors, in which case an error message is written.

All other arguments are passed through common blocks.

WRAPUP2 reads 12 scratch files, each of which contains the entire data series for one of the 12 data channels transcribed from a CGG formatted input file on the Mountain Optech optical disk. It reads the scratch files in parallel, piece by piece, and combines the 12 data streams into a sequence of ROSE formatted output files on VAX/VMS magnetic disk storage. Each resulting ROSE file contains data for a particular time period.

WRAPUP2 uses a table of starting and ending sample numbers to determine which samples to include in each successive ROSE output file. This table is created by the subroutine ROSETIME.

WRAPUP2 reads the first block of data from each scratch file and then calls the subroutine RDWRT to read, reformat, and write the remaining blocks.

WRAPUP2 calls HEADER to create and write ROSE file headers to each ROSE output file. It also calls HEADER2 to write the same header blocks to an alternate ROSE header file.

WRAPUP2 substitutes an alternate event date/time, or an alternate start date/time, or both, in output ROSE header in place of the date/time values derived from data in the originating CGG files, if either alternate date/time is included in the corresponding record in the date/time parameter file.

WRAPUP2 also optionally incorporates depth/range/bearing information from the date/time parameter file entries into the corresponding ROSE headers.

5.14 HEADER (NRG)

Subroutine HEADER creates and writes two (256 integer*2 word) ROSE headers. Some header fields are hardwired, such as sample rate (125), num-

ber of channels (12), number of scans per block (2048). Channel number, number of blocks, and length of last block are put into the file header, and the file header is written to the ROSE file.

The input arguments are:

integer nblks is number of blocks of data in the ROSE file
integer nleft is number of scans in last block

The output argument is:

integer istat returns 0 for success, -999 for failure.

Other values are passed in common files comrose.for and comio.for.

5.15 HEADER2 (WSL)

This subroutine modifies the ROSE header information previously established by HEADER. It changes the event number, the first sample time, the block count, and the last block size to correspond to ROSE files produced by program MODE 1 2, 3, or 4.

5.16 ROSETIME (WSL)

For each CGG input file, this subroutine is called by the main program to determine the starting and ending data points to be used for each of the (piecewise) ROSE output files.

Arguments are passed from ROSETIME to the calling program through common areas. The common area COMROSTIM contains the dates, times, starting and ending sample counts, the skip/save flags for each required ROSE output file, and flags indicating that the program has rounded or otherwise altered the requested ROSE date/times when calculating starting sample numbers.

ROSETIME returns ISTAT = 0 for normal completion, = -999 otherwise.

The routine reads a table of date/time values from FORTRAN I/O unit 55 (default file FOR055.DAT) and compares these values with those

contained in the header information of the corresponding CGG input file. It then calculates a relative time in milliseconds from the start of the data file to the start of the ROSE output file for this portion of the overall data series. It converts this time difference to an integer representing the number of the first sample to be placed in this ROSE output file. It then calculates the number of the last sample in this ROSE output file, by subtracting one from the number of the first sample in the next ROSE output file.

In converting date/time differences to sample numbers, the routine rounds to the nearest .008 seconds, which corresponds to a sampling rate of 125 samples per second. If the time difference between the CGG start time and the ROSE start time is .004, .005, .006, .007, .008, .009, .010, or .011 seconds, the routine rounds it to .008 seconds and corrects the ROSE start time and the starting sample count accordingly.

Accompanying each entry in the file of date/times is a "skip/save" code which allows the user to selectively write or not write the actual ROSE output file to disk storage. In MODE 1, if the skip/save code is a 1, the program will produce the ROSE file containing data for the requested time period. If the skip/save code is a 2, the program will complete all of the processing normally but will suppress the actual ROSE output file, thereby leaving a gap in the overall output data series. This feature is useful for processing CGG files in groups, as dictated by the priorities of the experiment. The 'sense' of the skip/save code values can be reversed by changing the CGG2ROSE2 operating mode from MODE 1 to MODE 2 so that a subsequent run of the program will create those ROSE output files with skip/save codes of 2 and omit those with skip/save codes of 1. The MODE 2 run thus produce the files to fill in the gaps left by the original MODE 1 run.

The file of date/time entries also contains the integer event numbers to be assigned by the user to the first ROSE output file created from each CGG input file. Successive ROSE output files for the same CGG input file are assigned incrementally increasing event numbers. Missing files caused by using the skip/save code to omit certain ROSE output files are still included in the numbering scheme, so that properly ordered event numbers can be assigned to these files later, should they be created by a subsequent program run.

The routine makes modifications to the date/time values supplied in the user's input table in order to insure that (1) there is at least one sample in each ROSE output file, and (2) the first ROSE output file for each CGG input file begins with sample number 1. Note that this means the first date/time value supplied for each CGG input file is actually a dummy value; it is ignored and replaced by the start date/time found in the CGG file header.

ROSETIME also reads optional alternate event date/times, alternate start date/times, depths, ranges, and bearings for each specified ROSE output file. The two alternate date/time values are used to override date/time values which are normally calculated from information in the originating CGG files.

The routine writes a list of CGG input files which are processed, together with the corresponding ROSE output files created from each CGG file. Files whose date/time values have been altered by the program are marked with an asterisk.

5.17 CALCDATE, CALCTIME, CALCYMD, ADDMIL, ACCUM (WSL)

Collectively, these subprograms are used by ROSETIME to convert date and time information from Gregorian to Julian format and vice versa, and to increment a date/time value by a given number of milliseconds. (See in-line program documentation for more thorough explanations of these subprograms.)

6 PROGRAM OPERATION

The source code for CGG2ROSE2 is contained in the following files:

CGG2ROSE2, TOMROSETIME, TOMLFASINPUTS, TOMLFASOUTPUTS, CGGVAX2

6.1 Compilation

After copying the source files to the current VAX/VMS login directory, they may be compiled in the normal manner:

```
$FORTRAN CGG2ROSE2
$FORTRAN TOMROSETIME
$FORTRAN TOMLFASINPUTS
$FORTRAN TOMLFASOUTPUTS
$FORTRAN CGGVAX2
```

6.2 Linking

The executable program CGG2ROSE2 may then be created using the VAX/VMS command

```
@CGG2ROSE2.LNK
```

which executes the following VAX/VMS command:

```
$LINK/EXE=CGG2ROSE2.EXE CGG2ROSE2,-
TOMLFASINPUTS,-
TOMLFASOUTPUTS,-
TOMROSETIME,-
CGGVAX2,-
SNS1:[SECURE]SLIB/LIB,-
SNS1:[OPTECH]OPTOLIB/LIB,-
CLINK/OPT
```

Here, SNS1: is name of the disk drive currently being used for the WHOI G & G Department software library [SECURE]SLIB and the Mountain Optech optical disk subroutine library [OPTECH]OPTOLIB.

The CLINK/OPT option causes the LINK options in the file CLINK.OPT to be included when creating the executable program. The CLINK.OPT file contains the option

```
SYS$LIBRARY:VAXCTRL.EXE/SHAREABLE
```

This links the Shareable VAX C Language Run Time Library along with the other object files and library modules. This option is required because the Mountain Optech subroutines are written in C.

(Prior to linking in the above manner, the two files CGG2ROSE2.LNK and CLINK.OPT should be placed in the current login directory.)

6.3 Execution

The CGG2ROSE2 program may be run using the following commands:

```
$MOUNT/FOREIGN DUC0: SNS4 SNS4
$ASSIGN file1 FOR055
$ASSIGN file2 FOR088
$RUN CGG2ROSE2
```

where file1 and file2 are the ROSE Date/Time Parameter File and the CGG Filename Parameter File respectively. DUC0: and SNS4 refer to the Mountain Optech optical disk device name and logical name.

The Primary Parameter File name does not need a \$ASSIGN command; it must be named CGG2ROSE2.PAR and must reside in the current login directory.

The program executes in the current login directory, but the directory which receives the ROSE output files is determined by information contained in the Primary Parameter File, for example, SNS2:[MICH.66]. This directory must be created prior to program execution, and must have the correct write access privileges. (Generally, a pair of directories, for example SNS2:[MICH.66] and SNS2:[MICH.67], must be created for a single run. See Section 4.7 for more information on directory naming conventions.) New directories are created the \$CREATE/DIR command, for example:

```
$CREATE/DIR [MICH.66]
$CREATE/DIR [MICH.67]
```

7 RESTRICTIONS

1. Minimum number of disk files opened simultaneously: 25
2. Minimum disk requirements:

Space for scratch files equal to size of largest CGG file to be processed, plus space for all requested ROSE files, plus 10% for directory overhead.

3. Maximum CGG input file allowed:

Determined by available scratch space and ROSE file space.

4. Restrictions on information in file of ROSE first sample dates/times:

Maximum of about 10 days between first sample date/time in CGG file and any sample date/time specified by user in the ROSE Date/Time Parameter File; otherwise an integer overflow may occur.

5. Maximum number of samples allowed in CGG input file:

There is no limit to the size of each CGG input file, except that there must be enough space on magnetic disk for scratch files and ROSE output. However, many of the programs that are used to process the resulting ROSE files (e.g., the plotting program LFAZO, the MICHUTIL dump/plot utility, and the library routine NEXTCHAN) can only process limited numbers of data points. For this reason CGG2ROSE2 should be used to split up large CGG files into ROSE files of manageable size, such as 6 minute files which contain 45000 samples (at the sampling rate of 125 samples/sec).

6. Illegal date values:

The program does not detect all illegal dates in the ROSE Date/Time Parameter File. For example, the illegal date 2/31/89 will not be detected.

The program only works between 1940 and 2039 because leap years are not handled correctly outside this period.

The program is not very forgiving of errors in the format of the ROSE Date/Time Parameter File such as incorrect record counts. Therefore, before beginning a long program run, the user should carefully check the contents of the file using the auxiliary program CKROSTIM (see Appendix 5).

7. Conflicting .LOG and .HDR file names:

There is a slight possibility that two or more CGG files will lead to identical log file and alternate header names. This occurs when two CGG files are created within the same 10 second period.

8. Maximum number of ROSE output files from a single CGG input file:
1000

9. Deployment number in the Primary Parameter File must be an even number.

This number must match the first of the two subdirectories which must be previously established for purposes of receiving the ROSE output files.

8 RECOMMENDED IMPROVEMENTS

1. Add code to skip over input blocks quickly by not writing them to the scratch files if they aren't needed for any ROSE output files.
2. For all except very large CGG input files, do the entire processing in memory to avoid the need for scratch files entirely.
3. Speed up the Mountain Optech open and read operations if possible.
4. Clean up the output formatting and logging by making more consistent use of logical unit numbers, format types, variable names, etc.
5. Simplify program input by combining input parameter files to eliminate redundancy and confusion.
6. Add another program MODE for verifying the content of the file of ROSE starting dates and times. (This exists at present as a separate program called CKROSTIM.)
7. Simplify or replace the subroutines which are used to manipulate date/time information.
8. Eliminate "dead" (unused) code left over from previous versions.

ACKNOWLEDGEMENT

We are indebted to the many individuals who participated in the LFASE experiment, as well as those who assisted in the creation of the CGG2ROSE2 program and the program report. Specifically, we would like to thank Steve Swift, Rob Goldsborough, Cleo Zani, and Don Koelsch for their assistance, cooperation, and understanding during this enterprise. Principal investigator Ralph Stephen deserves our particular thanks for his strong support of our efforts. His patience and understanding while managing LFASE software development efforts are to be commended.

This work was funded by the Office of Naval Research (Contract Number N00014-89-C-0018) and the Office of Naval Technology (Contract Number N00014-89-C-0179).

REFERENCES

- A. Compagnie Générale de Géophysique, 1988a. Multilock - Interface Unit - WHOI Version CGG, 1 rue Léon Migaux - BP 56, 91301, Massay, France. Ref. 64W03.
- B. Koelsch, D. E., Goldsborough, R. G., Berteaux, H. and Stephen R. A., 1990. A multi-node three component seismic system for Deep Sea Drilling Project (DSDP) boreholes. Proceedings of the Marine Technology Society Marine Instrumentation '90 meeting, San Diego, California.
- C. Latraille, S. L., 1983. Archiving and exchange of a computerized marine seismic database: the ROSE data archive system. Hawaii Institute of Geophysics, HIG-83-3.
- D. Latraille, S. L. and Dorman, L. M., 1983. A standard format for storage and exchange of natural and explosive-source seismic data. The ROSE format. Marine Geophysical Researches, 6, 99-105.
- E. Stephen, R. A., Koelsch, D. E., Bibee, D., Farrell, W. E., Pattee, W., and Spiess, F. N.. A borehole seismic array in the deep sea floor. Fall AGU, San Francisco, 1989, EOS, 70, 1216.

Appendices

A1. SAMPLE RUNS

- A1.1 Primary Parameter File (MODE 2)
- A1.2 CGG Filename Parameter File
- A1.3 ROSE Date/Time Parameter File (without alternate header information)
- A1.4 ROSE Date/Time Parameter File (with alternate header information)
- A1.5 Listing File (MODES 1, 2, and 3)
- A1.6 Log File (MODE 2)
- A1.7 Primary Parameter File (MODE 4)
- A1.8 Start and Event Time Listing (MODE 4)
- A1.9 Log File (MODE 4)

A2. Parameter File FORMATS and READ Statements

- A2.1 Primary Parameter File FORMATS and READ Statements
- A2.2 CGG Filename Parameter File FORMATS and READ Statements
- A2.3 ROSE Date/Time Parameter File FORMATS and READ Statements

A3. CGG File Structure

- A3.1 File and Record Formats for LFASE CGG Data on Optical Disk
- A3.2 CGG Numeric Data Notation

A4. WHOI ROSE File Format

- A4.1 ROSE Header Record and Data Structure
- A4.2 ROSE FILE Structure Summary

A5. Related Programs

A6. CGG2ROSE2 Program History

Appendix 1: Sample Runs

A1.1 Primary Parameter File (MODE 2)

01192140, 109	!	FIRST CGG FILE
01211502, 269	!	LAST CGG FILE
8486	!	FIRST SHOT IN SEQUENCE (DEFALUT EVENT NUMBER)
SNS2	!	DEVICE
MICH	!	EXPERIMENT
99	!	DEPLOYMENT NUMBER
01	!	RECEIVER
SKI	!	EXTENT
12,0	!	KILOMETER
13,0	!	METER RANGE
19,50	!	DEPTH
35,6	!	SOURCE
42,127	!	AZIM FULL DEGREE
43,0	!	AZIM HUNDRETHS
-99,2	!	EOF, MODE

↑
MODE 2

} ←
Optional values to be substituted
into header fields.

A1.2 CGG Filename Parameter File

01192140.109
01192141.419
01192142.179
01201915.389
01201935.389
01211242.599
01211244.319
01211245.599
01211247.299
01211458.179
01211502.269

A1.3 ROSE Date/Time Parameter File (without alternate header information)

01,5472	←	[number of ROSE output files for this CGG input file
2,01,19,89,21,40,09,000		
01,5473		[event number for first ROSE file
2,01,19,89,21,41,40,000		
01,5474		[skip/save flag
2,01,19,89,21,41,16,000		
06,5492		[month, day, year, hour, minute, second, millisecond for start of this ROSE file
2,01,20,89,19,20,00,008		
2,01,20,89,19,21,00,008		
1,01,20,89,19,22,01,000		
2,01,20,89,19,23,00,008		
1,01,20,89,19,24,00,000		
2,01,20,89,19,25,00,000		
06,6493		
1,01,20,89,19,40,00,000		
2,01,20,89,19,41,01,008		
2,01,20,89,19,42,00,008		
2,01,20,89,19,43,00,000		
2,01,20,89,19,44,02,016		
1,01,20,89,19,45,00,000		
01,5525		
2,01,21,89,12,43,24,000		
01,5526		
2,01,21,89,12,44,56,008		
01,5527		
2,01,21,89,12,46,24,000		
01,5528		
2,01,21,89,12,47,53,008		
01,5566		
1,01,21,89,14,58,17,008		
01,5567		
2,01,21,89,15,02,25,000		

Al.4 ROSE Date/Time Parameter File (with alternate header information)

(Note: This sample input file does not correspond to the same computer run as the other input and output files shown in this Appendix.)

```

007,0080 08111758.479
1 08,11,89,17,59, 2.000 08,11,89,17,59, 2.269 08,11,89,17,59, 2.269 5 0.
000 0.00 0.00 0.0 0.000 0.000 0.000 0.000 0
1 08,11,89,18,09, 2.000 08,11,89,18,09, 2.269 08,11,89,18,09, 2.269 6 0.
000 0.00 0.00 0.0 0.000 0.000 0.000 0.000 0
1 08,11,89,18,19, 2.000 08,11,89,18,19, 2.269 08,11,89,18,19, 2.269 6 0.
000 0.00 0.00 0.0 0.000 0.000 0.000 0.000 0
1 08,11,89,18,29, 2.000 08,11,89,18,29, 2.269 08,11,89,18,29, 2.269 6 0.
000 0.00 0.00 0.0 0.000 0.000 0.000 0.000 0
1 08,11,89,18,41,26.213 08,11,89,18,41,26.482 08,11,89,18,41,17.874 7 0006E 51.
647 179.48 25.00 67.5 0.164 0.170 28.809 -75.377 1
1 08,11,89,18,46,25.825 08,11,89,18,46,26.094 08,11,89,18,46,17.616 10 0007E 50.
865 179.71 25.00 71.7 0.157 0.166 28.802 -75.379 1
1 08,11,89,18,51,25.499 08,11,89,18,51,25.768 08,11,89,18,51,17.477 10 0008E 49.
745 179.49 25.00 66.4 0.166 0.168 28.792 -75.377 1
006,0100 08112000.309/
1 08,11,89,20,00,44.000 08,11,89,20,00,44.269 08,11,89,20,01,26.705 4 0022E 36.
972 178.59 25.00 80.6 0.144 0.192 28.677 -75.372 1
1 08,11,89,20,06,33.751 08,11,89,20,06,34.020 08,11,89,20,06,28.013 10 0023E 36.
041 178.37 25.00 74.9 0.152 0.191 28.668 -75.371 1
1 08,11,89,20,11,31.994 08,11,89,20,11,32.263 08,11,89,20,11,26.414 10 0024E 35.
093 178.17 25.00 79.9 0.145 0.192 28.660 -75.370 1
1 08,11,89,20,16,32.914 08,11,89,20,16,33.183 08,11,89,20,16,27.492 10 0025E 34.
147 178.00 25.00 77.0 0.149 0.193 28.651 -75.369 1
1 08,11,89,20,21,36.480 08,11,89,20,21,36.749 08,11,89,20,21,31.213 10 0026E 33.
218 177.80 25.00 67.5 0.164 0.194 28.643 -75.369 1
1 08,11,89,20,31,31.741 08,11,89,20,31,32.010 08,11,89,20,31,26.789 10 0028E 31.
327 177.40 25.00 0.0 0.000 0.000 28.626 -75.367 1

```

number of ROSE output files for
this CGG input file

event number for first ROSE file

CGGG input file name

shot instant information:

range
bearing
weight
depth
bubble pulse time
flight time
latitude
longitude
type of source (1=active,0=passive)

skip/save flag

date/time (month, day, year, hour,
minute, second, millisecond)
for start of this ROSE file

alternate start-of-data date/time
for ROSE header

alternate event date/time
for ROSE header

A1.5 Listing File (FOR018.DAT) (MODEs 1,2 or 3)

```

***** RUN OF CGG2ROSE2 AT 18-JUL-89 ON 13:27:53 *****
EVT MODAHRMN SECMIL BK LEFT FSTSMP LSTSMP OPTICALFILE DAHRMN SECMIL BK LEFT SC
5472 1192140 09.000 1 624 1 624 01192140.109 192140 31.000 1 624 A
5473 1192141 40.000 1 622 1 622 01192141.419 192142 02.000 1 622 A
5474 1192142 16.000 1 624 1 624 01192142.179 192142 37.000 1 624 A
5492 1201920 00.000 4 1357 1 7501 01201915.389 201920 00.000 22 1477 A
5493 1201921 00.008 4 1480 7502 15125 01201915.389 201920 00.000 22 1477 A
5495 1201923 00.008 4 1355 22502 30000 01201915.389 201920 00.000 22 1477 A
5497 1201925 00.000 4 841 37501 44485 01201915.389 201920 00.000 22 1477 A
6494 1201941 01.008 4 1231 7627 15001 01201935.389 201940 00.000 22 1477 A
6495 1201942 00.008 4 1355 15002 22500 01201935.389 201940 00.000 22 1477 A
6496 1201943 00.000 4 1608 22501 30252 01201935.389 201940 00.000 22 1477 A
6497 1201944 02.016 4 1104 30253 37500 01201935.389 201940 00.000 22 1477 A
5525 1211243 24.000 1 1241 1 1241 01211242.599 211243 24.000 1 1241 A
5526 1211244 56.000 1 1242 1 1242 01211244.319 211244 56.000 1 1242 A
5527 1211246 24.000 1 1246 1 1246 01211245.599 211246 24.000 1 1246 A
5528 1211247 54.000 1 1247 1 1247 01211247.299 211247 54.000 1 1247 A
5567 1211502 25.000 1 623 1 623 01211502.269 211502 52.000 1 623 A

```

A1.6 Log File (MODE 2) (partial listing)

(note: the scan counter error messages are written to logical unit 56
in the most recent version of the program.)

_DUB0:[MICH.WSL1]R19E21401.LOG;52

18-JUL-1989

input field	value
12	0
13	0
19	50
35	6
42	127
43	0
-99	2

*****

THE INPUT FILE WILL BE: 01192140.109

drive duc0: infile 01192140.109

	0	1	0	0	0	0	0
hed/trl w.r.c. clock	0	1	0	0	0	0	0
hed/trl dos clock	89	1	19	21	40	9	0
hed/trl w.r.c. start clock	0	1	0	0	0	0	0
hed/trl w.r.c. stop clock	0	1	0	0	0	0	0
hed/trl shot clock	0	1	0	0	0	0	0
selected dos clock for data start							
selected dos clock for event							
start date	89	1	19	21	40	9	0
event date	89	0	19	21	40	9	0

1	2	1	10000000	1	19	89	21	40	9	0
scan cnt err: samp no=	1	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	2	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	3	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	4	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	5	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	6	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	7	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	8	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	9	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	10	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	11	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	12	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	13	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	14	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	15	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	16	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	17	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	18	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	19	last cnt=	0	this cnt=	0					
scan cnt err: samp no=	20	last cnt=	0	this cnt=	0					

These error
messages will
normally not
appear when
the scan
counter is
working
properly.

(The scan
counter field
was set to
zero in the
CGG files used
in this sample
run.)

wrote out 624 scans

going back for trailer

	0	1	0	0	0	0	0
trailer scan count =	624						
hed/trl w.r.c. clock	0	1	0	0	0	0	0
hed/trl dos clock	89	1	19	21	40	31	0
hed/trl w.r.c. start clock	0	1	0	0	0	0	0
hed/trl w.r.c. stop clock	0	1	0	0	0	0	0
hed/trl shot clock	0	1	0	0	0	0	0
selected dos clock for data start							
selected dos clock for event							

*****

THE INPUT FILE WILL BE: 01192141.419

```

drive duc0:  infile 01192141.419
hed/trl w.r.c. clock      0      1      0      0      0      0      0
hed/trl dos clock      89      1      19      21      41      40      0
hed/trl w.r.c. start clock 0      1      0      0      0      0      0
hed/trl w.r.c. stop clock 0      1      0      0      0      0      0
hed/trl shot clock      0      1      0      0      0      0      0
selected dos clock for data start
selected dos clock for event
start date      89      1      19      21      41      40      0
event date      89      0      19      21      41      40      0
1 2      1      10000000 1 19 89 21 41 40 0
scan cnt err: samp no= 1 last cnt= 0 this cnt= 0
scan cnt err: samp no= 2 last cnt= 0 this cnt= 0
scan cnt err: samp no= 3 last cnt= 0 this cnt= 0
scan cnt err: samp no= 4 last cnt= 0 this cnt= 0
scan cnt err: samp no= 5 last cnt= 0 this cnt= 0
scan cnt err: samp no= 6 last cnt= 0 this cnt= 0
scan cnt err: samp no= 7 last cnt= 0 this cnt= 0
scan cnt err: samp no= 8 last cnt= 0 this cnt= 0
scan cnt err: samp no= 9 last cnt= 0 this cnt= 0
scan cnt err: samp no= 10 last cnt= 0 this cnt= 0
scan cnt err: samp no= 11 last cnt= 0 this cnt= 0
scan cnt err: samp no= 12 last cnt= 0 this cnt= 0
scan cnt err: samp no= 13 last cnt= 0 this cnt= 0
scan cnt err: samp no= 14 last cnt= 0 this cnt= 0
scan cnt err: samp no= 15 last cnt= 0 this cnt= 0
scan cnt err: samp no= 16 last cnt= 0 this cnt= 0
scan cnt err: samp no= 17 last cnt= 0 this cnt= 0
scan cnt err: samp no= 18 last cnt= 0 this cnt= 0
scan cnt err: samp no= 19 last cnt= 0 this cnt= 0
scan cnt err: samp no= 20 last cnt= 0 this cnt= 0
wrote out      622 scans
going back for trailer
trailer scan count =      622
hed/trl w.r.c. clock      0      1      0      0      0      0      0
hed/trl dos clock      89      1      19      21      42      2      0
hed/trl w.r.c. start clock 0      1      0      0      0      0      0
hed/trl w.r.c. stop clock 0      1      0      0      0      0      0
hed/trl shot clock      0      1      0      0      0      0      0
selected dos clock for data start
selected dos clock for event

```

*****

THE INPUT FILE WILL BE: 01192142.179

```

drive duc0:  infile 01192142.179
hed/trl w.r.c. clock      0      1      0      0      0      0      0
hed/trl dos clock      89      1      19      21      42      16      0
hed/trl w.r.c. start clock 0      1      0      0      0      0      0
hed/trl w.r.c. stop clock 0      1      0      0      0      0      0
hed/trl shot clock      0      1      0      0      0      0      0
selected dos clock for data start
selected dos clock for event

```

start date	89	1	19	21	42	16	0
event date	89	0	19	21	42	16	0
1 2	1	10000000	1 19	89 21	42 16	0 *	
scan cnt err: samp no=	1	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	2	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	3	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	4	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	5	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	6	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	7	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	8	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	9	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	10	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	11	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	12	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	13	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	14	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	15	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	16	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	17	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	18	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	19	last cnt=	0	this cnt=	0		
scan cnt err: samp no=	20	last cnt=	0	this cnt=	0		

wrote out 624 scans

going back for trailer

trailer scan count = 624

hed/trl w.r.c. clock	0	1	0	0	0	0	0

hed/trl dos clock	89	1	19	21	42	37	0

hed/trl w.r.c. start clock	0	1	0	0	0	0	0

hed/trl w.r.c. stop clock	0	1	0	0	0	0	0

hed/trl shot clock	0	1	0	0	0	0	0

selected dos clock for data start

selected dos clock for event

An asterisk indicates that the program has used a different date/time value from the one provided by the user in the ROSE Date/time Parameter File (in this case, because the first ROSE file must start with sample 1).

*****

THE INPUT FILE WILL BE: 01201915.389

drive duc0: infile 01201915.389

hed/trl w.r.c. clock	89	1	20	19	15	18	0

hed/trl dos clock	89	5	3	4	8	2	0

hed/trl w.r.c. start clock	89	1	20	19	20	0	0

hed/trl w.r.c. stop clock	89	1	20	19	26	0	0

hed/trl shot clock	0	1	0	0	0	0	0

selected w.r.c. start for data start

selected w.r.c. start for event time

start date	89	1	20	19	20	0	0

event date	89	0	20	19	20	0	0

1 2	1	7501	1 20	89 19	20 0	0 *	

2 2	7502	15125	1 20	89 19	21 0	8	

3 1	15126	22501	1 20	89 19	22 1	0	

4 2	22502	30000	1 20	89 19	23 0	8	

5 1	30001	37500	1 20	89 19	24 0	0	

6 2	37501	10000000	1 20	89 19	25 0	0	

scan cnt err: samp no=	1	last cnt=	0	this cnt=			

scan cnt err: samp no=	2	last cnt=	0	this cnt=			

scan cnt err: samp no=	3	last cnt=	0	this cnt=			

scan cnt err: samp no=	4	last cnt=	0	this cnt=	0
scan cnt err: samp no=	5	last cnt=	0	this cnt=	0
scan cnt err: samp no=	6	last cnt=	0	this cnt=	0
scan cnt err: samp no=	7	last cnt=	0	this cnt=	0
scan cnt err: samp no=	8	last cnt=	0	this cnt=	0
scan cnt err: samp no=	9	last cnt=	0	this cnt=	0
scan cnt err: samp no=	10	last cnt=	0	this cnt=	0
scan cnt err: samp no=	11	last cnt=	0	this cnt=	0
scan cnt err: samp no=	12	last cnt=	0	this cnt=	0
scan cnt err: samp no=	13	last cnt=	0	this cnt=	0
scan cnt err: samp no=	14	last cnt=	0	this cnt=	0
scan cnt err: samp no=	15	last cnt=	0	this cnt=	0
scan cnt err: samp no=	16	last cnt=	0	this cnt=	0
scan cnt err: samp no=	17	last cnt=	0	this cnt=	0
scan cnt err: samp no=	18	last cnt=	0	this cnt=	0
scan cnt err: samp no=	19	last cnt=	0	this cnt=	0
scan cnt err: samp no=	20	last cnt=	0	this cnt=	0

wrote out 1477 scans

going back for trailer

trailer scan count = 44485

hed/trl w.r.c. clock	89	1	20	19	32	32	0
hed/trl dos clock	89	6	13	6	44	31	0
hed/trl w.r.c. start clock	89	1	20	19	20	0	0
hed/trl w.r.c. stop clock	89	1	20	19	26	0	0
hed/trl shot clock	0	1	0	0	0	0	0

selected w.r.c. start for data start

elected w.r.c. start for event time

*****

A1.7 Primary Parameter File (MODE 4)

01192140, 109	!	FIRST CGG FILE
01211502, 269	!	LAST CGG FILE
8486	!	FIRST SHOT IN SEQUENCE (DEFALUT EVENT NUMBER)
SNS2	!	DEVICE
MICH	!	EXPERIMENT
99	!	DEPLOYMENT NUMBER
01	!	RECEIVER
SKI	!	EXTENT
12,0	!	KILOMETER
13,0	!	METER RANGE
19,50	!	DEPTH
35,6	!	SOURCE
42,127	!	AZIM FULL DEGREE
43,0	!	AZIM HUNDRETHS
-99,4	!	EOF, MODE

MODE 4

A1.8 Start and Event Time Listing (FOR045.DAT) (MODE 4)

```

***** RUN OF CGG2ROSE2 AT 18-JUL-89 ON 14:04:41 *****
CGG INPUT FILENAME FIRST SAMPLE TIME(SOURCE) EVENT TIME(SOURCE)
01192140.109 SAMP(D): 89 01 19 21 40 09.000 EVNT(D): 89 01 19 21 40 09.000
01192141.419 SAMP(D): 89 01 19 21 41 40.000 EVNT(D): 89 01 19 21 41 40.000
01192142.179 SAMP(D): 89 01 19 21 42 16.000 EVNT(D): 89 01 19 21 42 16.000
01201915.389 SAMP(W): 89 01 20 19 20 00.000 EVNT(W): 89 01 20 19 20 00.000
01201935.389 SAMP(W): 89 01 20 19 40 00.000 EVNT(W): 89 01 20 19 40 00.000
01211242.599 SAMP(W): 89 01 21 12 43 24.000 EVNT(W): 89 01 21 12 43 24.000
01211244.319 SAMP(W): 89 01 21 12 44 56.000 EVNT(W): 89 01 21 12 44 56.000
01211245.599 SAMP(W): 89 01 21 12 46 24.000 EVNT(W): 89 01 21 12 46 24.000
01211247.299 SAMP(W): 89 01 21 12 47 54.000 EVNT(W): 89 01 21 12 47 54.000
01211458.179 SAMP(D): 89 01 21 14 58 17.000 EVNT(D): 89 01 21 14 58 17.000
01211502.269 SAMP(D): 89 01 21 15 02 25.000 EVNT(D): 89 01 21 15 02 25.000

```

Clock Code: D = DOS clock selected
 W = Webb (WRC) clock selected
 S = System clock selected
 N = No clock selected

18-JUL-1989

_DUB0:[MICH.WSL1]R19E21401.LOG;53

input field	value
12	0
13	0
19	50
35	6
42	127
43	0
-99	4

*****

THE INPUT FILE WILL BE: 01192140.109

drive duc0: infile 01192140.109							
hed/trl w.r.c. clock	0	1	0	0	0	0	0
hed/trl dos clock	89	1	19	21	40	9	0
hed/trl w.r.c. start clock	0	1	0	0	0	0	0
hed/trl w.r.c. stop clock	0	1	0	0	0	0	0
hed/trl shot clock	0	1	0	0	0	0	0
selected dos clock for data start							
selected dos clock for event							
start date	89	1	19	21	40	9	0
event date	89	0	19	21	40	9	0

*****

THE INPUT FILE WILL BE: 01192141.419

drive duc0: infile 01192141.419							
hed/trl w.r.c. clock	0	1	0	0	0	0	0
hed/trl dos clock	89	1	19	21	41	40	0
hed/trl w.r.c. start clock	0	1	0	0	0	0	0
hed/trl w.r.c. stop clock	0	1	0	0	0	0	0
hed/trl shot clock	0	1	0	0	0	0	0
selected dos clock for data start							
selected dos clock for event							
start date	89	1	19	21	41	40	0
event date	89	0	19	21	41	40	0

*****

THE INPUT FILE WILL BE: 01192142.179

drive duc0: infile 01192142.179							
hed/trl w.r.c. clock	0	1	0	0	0	0	0
hed/trl dos clock	89	1	19	21	42	16	0
hed/trl w.r.c. start clock	0	1	0	0	0	0	0
hed/trl w.r.c. stop clock	0	1	0	0	0	0	0
hed/trl shot clock	0	1	0	0	0	0	0
selected dos clock for data start							
selected dos clock for event							
start date	89	1	19	21	42	16	0
event date	89	0	19	21	42	16	0

Appendix 2: Parameter File Formats (and READ statements)

A2.1 Primary Parameter File FORMATS and READ Statements:

```

c  punit = 20
   read (punit,*,err = 30) ISTNUM,ISTYR
   READ (PUNIT,*,ERR=30) IENNUM,IENYR
   READ (PUNIT,*) ISHOT_ST

   READ (PUNIT,9000,ERR=30) KDEV
9000 FORMAT (A)

   READ (PUNIT,9000,ERR=30) KCRUISE
   READ (PUNIT,*,ERR=30) ichangDEPL
   READ (PUNIT,*,ERR=30) NNIID
   READ (PUNIT,9000,ERR=30) KEXTENT

   read(punit,*,end = 100,err = 50) indval,ival

```

A2.2 CGG Filename File FORMATS and READ Statements:

```

DO IP=1,1000

   READ (88,5000,END=5100) IFILEMONTH(IP),
*                               IFILEDAY(IP),IFILEHRMN(IP),
*                               IFILESEC(IP),IFILEYEAR(IP)

5000   FORMAT (1X,2(I2),I4,1X,I2,I1)

END DO

```

A2.3 ROSE Start Date/Time Parameter File FORMATS and READ Statements:

```

   read(55,1100,end=96) nf,naltevent,thisoptfile

1100   FORMAT(5X,I3,1X,I4,1X,A12)

   read (55,200,end=96) isv(n),fmo(n),fda(n),fyr(n),
*                               fhr(n),fmin(n),fsec(n),fmil(n),
*                               altnsmo(n), altnsda(n), altnsyr(n),altnshr(n),
*                               altnsmin(n),altnssec(n),altnsmil(n),
*                               altnemo(n), altneda(n), altneyr(n),altnehr(n),
*                               altnemin(n),altnesec(n),altnemil(n),
*                               range(n),bear(n),wgt(n),sdepth(n),bubbl(n),
*                               flitime(n),alat(n),along(n),ikind(n)

200 format (1x,i1,3(6(1x,i2),1x,i3),
*          10x,f6.3,1x,f6.2,1x,f5.2,1x,f5.1,1x,f5.3,
*          1x,f5.3,1x,f6.3,1x,f7.3,1x,i1)

```

Appendix 3: CGG File Structure

A3.1 File and record formats for LFASE CGG data on Optical Disk

by A.C.Zani and N.Galbraith 89/06/19

The files are composed of a header record, a series of data records, and a trailer record. This documents the format of these records as of May 04 1989. Structure and variable names are presented in quotes and may be found in the C source file struct.h on the MS-DOS based logging computer.

A CGG data file for the LFASE experiment is comprised of:

I) HEADER RECORD

The header is an "ENG_DATA" structure of 256 bytes containing:
Offset 0, 4 bytes, "header_sync" contains Header sync: BFFFC000.

followed by

a "CLOCKS" structure of 120 bytes containing:

vv a "wrc" MTIME structure of 20 bytes
containing the WRC clock at file time arranged as:

Offset 4, 2 bytes, "milli" contains milliseconds.

and a "tm" TIME structure of 18 bytes

containing the date and time arranged as:

Offset 6, 2 bytes, "tm_sec" contains seconds after the minute - [0,59].
Offset 8, 2 bytes, "tm_min" contains minutes after the hour - [0,59].
Offset 10, 2 bytes, "tm_hour" contains hours since midnight - [0,23].
Offset 12, 2 bytes, "tm_mday" contains day of the month - [1,31].
Offset 14, 2 bytes, "tm_mon" contains months since January - [0,11].
Offset 16, 2 bytes, "tm_year" contains years since 1900.
Offset 18, 2 bytes, "tm_wday" contains days since Sunday - [0,6].
Offset 20, 2 bytes, "tm_yday" contains days since January 1 - [0,365].
Offset 22, 2 bytes, "tm_isdst" contains daylight savings time flag.

followed by a "sys" MTIME structure of 20 bytes

containing the DOS clock at file time arranged as:

Offset 24, 2 bytes, "milli" contains milliseconds.

and a "tm" TIME structure of 18 bytes

containing the date and time arranged as:

Offset 26, 2 bytes, "tm_sec" contains seconds after the minute - [0,59].
Offset 28, 2 bytes, "tm_min" contains minutes after the hour - [0,59].
Offset 30, 2 bytes, "tm_hour" contains hours since midnight - [0,23].
Offset 32, 2 bytes, "tm_mday" contains day of the month - [1,31].
Offset 34, 2 bytes, "tm_mon" contains months since January - [0,11].
Offset 36, 2 bytes, "tm_year" contains years since 1900.
Offset 38, 2 bytes, "tm_wday" contains days since Sunday - [0,6].
Offset 40, 2 bytes, "tm_yday" contains days since January 1 - [0,365].
Offset 42, 2 bytes, "tm_isdst" contains daylight savings time flag.

followed by a "start" MTIME structure of 20 bytes

containing the CGG data start time arranged as:

Offset 44, 2 bytes, "milli" contains milliseconds.

and a "tm" TIME structure of 18 bytes

containing the date and time arranged as:

Offset 46, 2 bytes, "tm_sec" contains seconds after the minute - [0,59].
Offset 48, 2 bytes, "tm_min" contains minutes after the hour - [0,59].
Offset 50, 2 bytes, "tm_hour" contains hours since midnight - [0,23].
Offset 52, 2 bytes, "tm_mday" contains day of the month - [1,31].

Offset 54, 2 bytes, "tm_mon" contains months since January - [0,11].
 Offset 56, 2 bytes, "tm_year" contains years since 1900.
 Offset 58, 2 bytes, "tm_wday" contains days since Sunday - [0,6].
 Offset 60, 2 bytes, "tm_yday" contains days since January 1 - [0,365].
 Offset 62, 2 bytes, "tm_isdst" contains daylight savings time flag.

followed by a "stop" MTIME structure of 20 bytes
 containing the CGG data stop time arranged as:

Offset 64, 2 bytes, "milli" contains milliseconds.

and a "tm" TIME structure of 18 bytes
 containing the date and time arranged as:

Offset 66, 2 bytes, "tm_sec" contains seconds after the minute - [0,59].
 Offset 68, 2 bytes, "tm_min" contains minutes after the hour - [0,59].
 Offset 70, 2 bytes, "tm_hour" contains hours since midnight - [0,23].
 Offset 72, 2 bytes, "tm_mday" contains day of the month - [1,31].
 Offset 74, 2 bytes, "tm_mon" contains months since January - [0,11].
 Offset 76, 2 bytes, "tm_year" contains years since 1900.
 Offset 78, 2 bytes, "tm_wday" contains days since Sunday - [0,6].
 Offset 80, 2 bytes, "tm_yday" contains days since January 1 - [0,365].
 Offset 82, 2 bytes, "tm_isdst" contains daylight savings time flag.

followed by a "shot" MTIME structure of 20 bytes
 containing the KTT shot instant time arranged as:

Offset 84, 2 bytes, "milli" contains milliseconds.

and a "tm" TIME structure of 18 bytes
 containing the date and time arranged as:

Offset 86, 2 bytes, "tm_sec" contains seconds after the minute - [0,59].
 Offset 88, 2 bytes, "tm_min" contains minutes after the hour - [0,59].
 Offset 90, 2 bytes, "tm_hour" contains hours since midnight - [0,23].
 Offset 92, 2 bytes, "tm_mday" contains day of the month - [1,31].
 Offset 94, 2 bytes, "tm_mon" contains months since January - [0,11].
 Offset 96, 2 bytes, "tm_year" contains years since 1900.
 Offset 98, 2 bytes, "tm_wday" contains days since Sunday - [0,6].
 Offset 100, 2 bytes, "tm_yday" contains days since January 1 - [0,365].
 Offset 102, 2 bytes, "tm_isdst" contains daylight savings time flag.

followed by a "schedule" MTIME structure of 20 bytes
 containing the special wakeup time arranged as:

Offset 104, 2 bytes, "milli" contains milliseconds.

and a "tm" TIME structure of 18 bytes
 containing the date and time arranged as:

Offset 106, 2 bytes, "tm_sec" contains seconds after the minute - [0,59].
 Offset 108, 2 bytes, "tm_min" contains minutes after the hour - [0,59].
 Offset 110, 2 bytes, "tm_hour" contains hours since midnight - [0,23].
 Offset 112, 2 bytes, "tm_mday" contains day of the month - [1,31].
 Offset 114, 2 bytes, "tm_mon" contains months since January - [0,11].
 Offset 116, 2 bytes, "tm_year" contains years since 1900.
 Offset 118, 2 bytes, "tm_wday" contains days since Sunday - [0,6].
 Offset 120, 2 bytes, "tm_yday" contains days since January 1 - [0,365].
 Offset 122, 2 bytes, "tm_isdst" contains daylight savings time flag.

followed by a "PROGRAM_STATE" structure of 60 bytes containing:

a "failures" structure of 18 bytes

containing the cumulative system failures arranged as:

Offset 124, 2 bytes, "clock" contains WRC clock errors.
 Offset 126, 6 bytes, "recorder" contains optical disks 1,2,3 errors.
 Offset 132, 2 bytes, "state" contains state file read/write errors.
 Offset 134, 2 bytes, "memory" contains ported memory errors.
 Offset 136, 2 bytes, "system" contains program system (implies a bug).
 Offset 138, 2 bytes, "command" contains line noise & bad commands.
 Offset 140, 2 bytes, "conversion" contains A/D convertor timeouts.

followed by a "wakeups" structure of 14 bytes
containing the cumulative system wakeups arranged as:

Offset 142,	2 bytes, "clock" contains WRC clock.
Offset 144,	4 bytes, "xpdr" contains transponder 1, 2.
Offset 148,	2 bytes, "com" contains communications link.
Offset 150,	2 bytes, "mpl" contains MPL power up.
Offset 152,	2 bytes, "watchdog" contains hardware watchdog.
Offset 154,	2 bytes, "spurious" contains implied error or noise.

followed by a "next_wakeup" MTIME structure of 20 bytes
containing the next WRC wakeup arranged as:

Offset 156,	2 bytes, "milli" contains milliseconds.

and a "tm" TIME structure of 18 bytes
containing the date and time arranged as:

Offset 158,	2 bytes, "tm_sec" contains seconds after the minute - [0,59].
Offset 160,	2 bytes, "tm_min" contains minutes after the hour - [0,59].
Offset 162,	2 bytes, "tm_hour" contains hours since midnight - [0,23].
Offset 164,	2 bytes, "tm_mday" contains day of the month - [1,31].
Offset 166,	2 bytes, "tm_mon" contains months since January - [0,11].
Offset 168,	2 bytes, "tm_year" contains years since 1900.
Offset 170,	2 bytes, "tm_wday" contains days since Sunday - [0,6].
Offset 172,	2 bytes, "tm_yday" contains days since January 1 - [0,365].
Offset 174,	2 bytes, "tm_isdst" contains daylight savings time flag.
Offset 176,	6 bytes, "portmem" contains ported memory pointers.
Offset 182,	2 bytes, "recorder" contains which recorder in use.

followed by a "DIGITALS" structure of 14 bytes containing:

a "bites" structure of 8 bytes
containing the collected byte-wide status arranged as:

Offset 184,	1 bytes, "wrc_adjust" contains WRC adjustment from acqui end.
Offset 185,	1 bytes, "acq_mode" contains acquisition mode: 0-acqui, 1-pulse.
Offset 186,	1 bytes, "software" contains program: 1-bcup, 2-tcup, 3-test.
Offset 187,	1 bytes, "hardware" contains hardware: 1=XT, 2=AT.
Offset 188,	1 bytes, "version" contains software version.
Offset 189,	1 bytes, "spare3" contains spare byte.
Offset 190,	1 bytes, "spare2" contains spare byte.
Offset 191,	1 bytes, "spare1" contains spare byte.

a "bits" structure of 2 bytes containing the collected
bit-wide status arranged as:

Offset 192,	2 bytes, contains these status bits:
	clamp_alert clamp alert
	cgg_data_ok CGG data is active
	cgg_clk_ok CGG clock is active
	leak_bcu leak in the cpu ..(glug)
	leak_dru leak in the data recording unit
	mpl_pwr_det MPL power is on
	acqui_synchro acqui synchro is enabled
	ready ready status during power up
	clock_wakeup clock wakeup
	xponder1_wakeup transponder 1 wakeup
	xponder2_wakeup transponder 2 wakeup
	mpl_power_wakeup MPL power wakeup
	com_link_wakeup communications link wakeup
	watchdog_wakeup hardware watchdog wakeup
	spurious line noise?
	spare1 spare bit

and

Offset 194,	4 bytes, "scans" contains expected/actual scans in file.

followed by an "ANALOGS" structure of 32 bytes containing:

Offset 198, 2 bytes, "roll" contains instrument roll.
 Offset 200, 2 bytes, "pitch" contains instrument pitch.
 Offset 202, 2 bytes, "bat_p24v" contains battery +24 volts.
 Offset 204, 2 bytes, "mpl_p28v" contains MPL +28 volts.
 Offset 206, 2 bytes, "cpu_p5v" contains CPU +5 volts.
 Offset 208, 2 bytes, "cpu_pl2v" contains CPU +12 volts.
 Offset 210, 2 bytes, "cpu_ml2v" contains CPU -12 volts.
 Offset 212, 2 bytes, "bat_p5v" contains battery +5 volts.
 Offset 214, 2 bytes, "dru_p5v" contains DRU +5 volts.
 Offset 216, 2 bytes, "dru_pl2v" contains DRU +12 volts.
 Offset 218, 2 bytes, "spare6" contains spare a/d channel.
 Offset 220, 2 bytes, "spare5" contains spare a/d channel.
 Offset 222, 2 bytes, "spare4" contains spare a/d channel.
 Offset 224, 2 bytes, "spare3" contains spare a/d channel.
 Offset 226, 2 bytes, "spare2" contains spare a/d channel.
 Offset 228, 2 bytes, "spare1" contains spare a/d channel.

followed by an array of 26 bytes containing:
 Offset 230, 26 bytes, "padding" contains spare capacity.

followed by
 II) DATA RECORDS consist of some number of "SCAN" structures of 32 bytes arranged as:

Offset 256, 3 bytes, "sync" contains CGG sync: 3FFFC0.
 Offset 259, 1 bytes, "auxn" contains aux channel number.
 Offset 260, 12 bytes, "lchan" contains data channels 1-6.
 Offset 272, 2 bytes, "auxd" contains aux data.
 Offset 274, 2 bytes, "pad" contains zero padding.
 Offset 276, 12 bytes, "hchan" contains data channels 7-12.

followed by
 III) TRAILER RECORD which is identical in format to the header.

A3.2 CGG Numeric Data Notation

CGG data coming into the acquisition computer in CGG format will be written verbatim plus a header and trailer record will be written at the start and end of each data file. The data format is as documented in the Multilock Guide except that the 2 'extra bits' are written at the start, not the end of each record, to make decoding easier. The start of any data record will therefore be 3fffC0(hex).

CGG data record:
 16 - 16 bit words per logical record
 12 channels of data, 4 extra (and aux channel) words

CGG data word:
 4 bits of gain, sign bit, 11 bits of mantissa

The mantissa is stored in 2's complement form. The gain has a range of 0 to 11, as documented in the Multilock User's Guide. The data range is therefore between 0.0012 (mantissa = 1, octal gain = 11) and 4994.6802 (mantissa = 3777 octal gain = 0).

To denormalize a CGG data word:

$$V_{mx} = 2.44 * M / 2^{**G}$$

To mask the components of the data word, use the following:

sign mask	4000 OCTAL
mantissa mask	3777 OCTAL
gain mask	170000 OCTAL

Appendix 4: WHOI ROSE File Format

A4.1 ROSE Header Record and Data Structure (from LaTraille, 1983)

The data section consists of a sequence of files each of which contains its own "file header record" followed by a data record(s) for each component, or channel, in the instrument. Optional additional headers can be used for instruments with more than 10 channels. The header is in integer format because almost all the data are numerical, and each header is 256 words long.

Each data record is 4096 words long. This size was chosen because most minicomputers can accommodate it in their I/O buffers. If a particular event requires more than 4096 words, the first 4096-word record is followed immediately with records 2, 3, ... until the event is described (see Figure 1). All samples from component 1 are written before going onto component 2. Thus, all of the data for an event are contained within a single file. Each file contains a number of data records equal to the number of components times the number of data records per component. If a data record contains less than 4096 samples, the record is padded with zeros. Words #71 and #72 in the header record are particularly important. Word #71 specifies the number of records required to fully describe an event for one component; word #72 specifies the number of non-zero samples in the last record of that component. The data are in two's-complement 16-bit integer format. Scales have been chosen so that a 16-bit word does not overflow. The format can also accommodate 32-bit integer data words, and in this case each record would contain 2048 data words. The sensitivities of the instruments, the electronics gains and the value of the least significant bit are specified in the header in order to preserve quantitative signal levels. The contents of the file header record are described below and a sample is shown in Table 2.

<u>Words</u>	<u>Contents</u>
1	Instrument Identification number. (This should be the integer equivalent of the identification number on the tape header so an instrument will have an alphanumeric and an integer representation on the same tape.)
2	Code for earthquake or shot data. If earthquake, set = 1; if shot, set = 2.
3	Earthquake or shot number. Shots are given specific numbers during an experiment. Numbers for earthquakes are simply a convenience-- the true identifier for an earthquake is its time.
4	Year of first sample in file; 2-digit representation.
5	Month of first sample in file.
6	Day of first sample in file.
7	Hour of first sample in file.
8	Minute of first sample in file.

- 9   Seconds of first sample in file.
- 10   Millisecs of first sample in file. (All times UTC and corrected to WWV.)
- 11   Range in whole megameters between event and instrument.
- 12   Remainder of range in whole kilometers between event and instrument.
- 13   Remainder of range in whole meters between event and instrument.
- 14   Estimate of error in range in meters. As the method of computing the range may be subject to error and may vary according to method, information describing the calculation method and the sound speed data used is filed with the exchange center.
- 15   Instrument latitude in degrees.
- 16   Fractional part of instrument latitude in millidegrees.
- 17   Instrument longitude in degrees.
- 18   Fractional part of instrument longitude in millidegrees.
- 19   Instrument depth in meters.
- 20   Normal incidence (vertical) travel time from surface to instrument (millisecs).
- 21   Shot or earthquake latitude in degrees.
- 22   Fractional part of shot or earthquake latitude in millidegrees.
- 23   Shot or earthquake longitude in degrees.
- 24   Fractional part of shot or earthquake longitude in millidegrees.
- 25   Error radius in event location in meters. As the method of computing the event location may be subject to error and may vary according to the available information, e.g., satellite or LORAN for shots, some documentation describing the estimation method and the data used is filed with the data exchange center.
- 26   Shot depth in meters or epicenter depth in kilometers.
- 27   Water depth at shot or earthquake (in meters).
- 28   Year of shot or earthquake; 2-digit representation.
- 29   Month of shot or earthquake.
- 30   Day of shot or earthquake.

- 31 Hour of shot or earthquake.
- 32 Minute of shot or earthquake.
- 33 Second of shot or earthquake.
- 34 Millisec of shot or earthquake. (All times in UTC corrected to WWV.)
- 35 Code for explosive type used (1 = TOVEX, 2 = dynamite, 3 = tetratol, 4 = SUS (1.8 lbs.), 5 = maxipulse, 6 = airgun, 7 = T.N.T., 8 = open, 9 = other).
- 36 Shot weight in $1000 \log_{10} W$, where W is charge weight in grams; or 10M where M is the earthquake magnitude.
- 37 Bubble pulse period in millisecs.
- 38 Sampling rate in samples/sec.
- 39 Number of digitized components.
- 40 Number of data words in record (= 4096 for 16-bit words or 2048 for 32-bit words).
- 41 File number within tape.
- 42-59 Blank for additions and changes. ← [azimuth
hundredths of azimuth
- 60 Code for duplicate component parameters - 1 if all channels have same parameters as channel 1. If channel parameters individually coded, this code = 0.
- 61 Code for component number one, e.g., 1) vertical, 2) radial, 3) tangential, 4) hydrophone, 5) non-rotated first horizontal, 6) non-rotated second horizontal, 7) water wave channel, 8) time.
- 62 Azimuth of non-rotated first horizontal if 61 is (5) or (6).
- 63 Sensitivity of instrument at frequency f_0 . (For displacement instruments - milli volts/millimicron; for velocity instruments - milli volts/(cm/sec), or volts (m/sec), and for hydrophones - volts/micropascal. The detailed response curve of each instrument is on file with the exchange center.)
- 64 f_0 in milliHertz - the frequency at which the above sensitivity of the instrument was measured.
- 65 Lower cutoff of passband in instrument amplifier (Hz).
- 66 Upper cutoff of passband in instrument amplifier (Hz).

- 67   Amplifier gain of digitizing filter in dB.
- 68   Lower cutoff of passband of digitizing filter (Hz).
- 69   Upper cutoff of passband of digitizing filter (Hz).
- 70   Value of the low order, or least significant, bit of the 16- or 32-bit integer word in microvolts.
- 71   Number of records in component.
- 72   Number of samples in last record.
- 73-80   Blank for changes and additions.
- 81-100   Repeat content of words 61-80 for the second component.
- 101   Continue repeating at intervals of 20 words for each component in the instrument as required up to 10 components.

If channel parameters for more than ⁹ components (channels) are to be encoded, add as many additional 256-word records as needed (one for each 12 additional components). Encode the contents of words 61-80 for the ¹⁰ component in words 1-20 of the new header record. Continue repeating at intervals of 20 words for each additional component as required. Note: if additional event header records are used, there must be a 0 in word 60 of the main event header.

TABLE 2. Sample Contents of File Header Record

Word Number	Field Description	Value
1	Instrument I.D. Number.	5
2	Code for earthquake or shot data. If earthquake, set=1, if shot, set=2.	2
3	Earthquake or shot number.	3243
4	Year of first sample in file.	79
5	Month of first sample in file.	2
6	Day of first sample in file.	28
7	Hour of first sample in file.	0
8	Minute of first sample in file.	5
9	Second of first sample in file.	9
10	Millisecond of first sample in file.	310
11	Range in whole megameters (between event and instrument).	0
12	Remainder of range in whole kilometers.	24
13	Remainder of range in whole meters.	368
14	Estimate of error in range in meters.	500
15	Instrument latitude in degrees.	15
16	Fractional part of instrument latitude in millidegrees.	29
17	Instrument longitude in degrees.	-104
18	Fractional part of instrument longitude in millidegrees.	-750
19	Instrument depth in meters.	2476
20	Vertical travel time from surface to instrument (msec).	1692
21	Event latitude in degrees.	15
22	Fractional part of event latitude in millidegrees.	253
23	Event longitude in degrees.	-104
24	Fractional part of event longitude in millidegrees.	-756
25	Error radius in event location in meters.	0
26	Shot depth in meters; or epicenter depth in kilometers.	41
27	Water depth at event in meters.	2681
28	Year of shot or earthquake.	79
29	Month of shot or earthquake.	2
30	Day of shot or earthquake.	28
31	Hour of shot or earthquake.	0
32	Minute of shot or earthquake.	5
33	Second of shot or earthquake.	9
34	Millisecond of shot or earthquake.	342
35	Code for explosive type used (1 = TOVEX, 2 = dynamite, 3 = tetratol, 4 = SUS (1.8 lbs.), 5 = maxipulse, 6 = airgun, 7 = TNT, 8 = open, 9 = other).	3
36	Shot weight in $1000 \log_{10} W$, where W is charge weight in grams; or LOM where M is the earthquake magnitude.	3434
37	Bubble pulse period in milliseconds.	110
38	Sampling rate in samples/second.	100
39	Number of digitized components.	2
40	Number of data words per record.	4096
41	File number within tape.	2
42-59	Blank.	

A4.2 ROSE File Structure Summary

THIS FILE WILL SHOW ROSE HEADER INFORMATION
THE ROSE HEADER WORD NUMBER AND THE CONTENT IS GIVEN

ROSE HEADER	CONTENTS
3	SHOT NUMBER
4	FIRSTSAMPLE YEAR
5	FIRSTSAMPLE MONTH
6	FIRSTSAMPLE DAY
7	FIRSTSAMPLE HOUR
8	FIRSTSAMPLE MINUTE
9	FIRSTSAMPLE MILLISECOND ←10 FIRSTSAMPLE SECOND
11	RANGE MEGAMETERS
12	RANGE WHOLE KILOMETERS
13	RANGE REMAINDER IN WHOLE METERS
15	LATITUDE DEGREES INSTRUMENT
16	LATITUDE MILLIDEGREES INSTRUMENT
17	LONGITUDE DEGREES INSTRUMENT
18	LONGITUDE MILLIDEGREES INSTRUMENT
19	INSTRUMENT DEPTH METERS
21	LATITUDE DEGREES SHOT
22	LATITUDE MILLIDEGREES SHOT
23	LONGITUDE DEGREES SHOT
24	LONGITUDE MILLIDEGREES SHOT
26	SHOT DEPTH IN METERS
27	WATER DEPTH AT SHOT
28	EVENT YEAR
29	EVENT MONTH
30	EVENT DAY
31	EVENT HOUR
32	EVENT MINUTE
33	EVENT SECOND
34	EVENT MILLISECOND
35	SOURCE TYPE: 5 MAXIPULSE, 6 AIRGUN, 1 TOVEX
36	CHARGE WEIGHT
38	SAMPLE RATE
39	NUMBER OF CHANNELS
40	NUMBER OF WORDS/RECORD: 4094 OR 2048
42	BEARING TO SHOT FROM INSTRUMENT WHOLE DEGREES
43	BEARING TO SHOT FROM INSTRUMENT DECIMAL DEGREES
60	-1 IF ALL CHANNELS HAVE THE SAME PARAMETER, 0 IF NOT
61	COMPONENT NUMBER
71	NUMBER OF RECORDS
72	NUMBER OF SAMPLES IN LAST RECORD
61-80	FOR FIRST CHANNEL AND EVERY 20 THEREAFTER FOR EACH CHANNEL

Appendix 5: Related Programs

Several programs have been developed which are related to CGG2ROSE2. Brief descriptions of those programs used for data validity checking and data exchange are given here. Scientific analysis and display programs are beyond the scope of this report.

MICHUTIL

Program MICHUTIL reads the ROSE data files, converts the integer coded decimal values to VAX floating point, and prints the values.

ODUMP

Program odump reads a CGG file from optical disk and displays it in hexadecimal to the screen. No formatting is done. This program was written for diagnostics for the optical disk files.

CGGLIST

Program cgglst reads CGG files from optical disk and lists header and data to the screen. Header data is displayed as integer*2 where appropriate (data and time fields) and in hex where necessary.

CKROSTIM

Checks the time values in the ROSE Date/Time Parameter File for validity and chronological order.

ROSETOUT

Creates a magnetic tape copy of ROSE data files, for data exchange with other investigators. On output to tape, program ROSETOUT creates a tape header using hard-coded fields. The last block of each channel for each file is padded with zeros, and the integer fields are byte-swapped to lsb-msb order for IBM compatibility. Data on disk is assumed to be stored as 32-bit integer.

ROSETIN

Reads a ROSE format tape and creates disk files in VAX notation. On input from tape, program ROSETIN strips the padding characters from each channel, and the data is byte-swapped to msb-lsb order. Input data is copied to disk in either 16-bit or 32-bit integer.

Appendix 6: CGG2ROSE2 Program History

CGG2ROSE2 Version 1.1 is the fourth revision of the CGG-to-ROSE file conversion program. The first version CGG2ROSE was written by N. Galbraith for the Michigan land-based test experiment MICH, in preparation for the LFASE experiment. T. Bolmer created a modified version of CGG2ROSE, called CGG2ROSE1, during the MICH experiment in order to process more than one CGG input file during a single execution of the program. Following the MICH experiment, the program was revised by W. Little to allow for the creation of multiple piecewise ROSE output files from each CGG input file. Several other enhancements were added in preparation for the LFASE experiment and the program was renamed CGG2ROSE2 Version 1.0. Finally, following the successful ocean bottom testing and initial at-sea data conversion, further modifications were made by Bolmer in an effort to increase quality control and simplify program operation, thereby creating CGG2ROSE2 Version 1.1.

(CGG2ROSE2 Version 1.1 contains several new modules not present in earlier versions. These are WRAPUP2, HEADER2, ROSETIME, CALCDATE, CALCTIME, CALCYMD, ADDMIL, ACCUM, and the common area COMROSTIM. The main program CGG2ROSE2 is a major modification of the older main program CGG2ROSE1. Minor changes have been made to OPDRDHED, OPDRDTR, OPDRDAT, OUTOPEN, WRTOUT, and CGGVAX.)

The principle distinction between CGG2ROSE2 and its predecessors is its ability to divide each CGG input file into a series of ROSE output files, each covering only a portion of the time period covered by each original CGG file. This typically reduces each resulting ROSE file to a more manageable size. In particular, it allows certain other post-processing programs such as LFAZO (a version of the plotting program BONZO written for LFASE) and NEXTCHAN (the ROSE file channel extraction subprogram in the WHOI G&G Department software library) to function properly, since CGG files often contain time series whose lengths exceed the storage limits of these older programs.

The data acquisition programs which execute on the MS-DOS system

within the instrument package were developed by A. C. Zani, who also provided many ancillary programs for instrument diagnosis, on-board computer troubleshooting, and data preprocessing.