FINAL REPORT

Computer Programs to Translate LIS Format, Vertical Seismic Profile (VSP) Data to SEG-Y Format Exchange Data Tapes

Mark Wiederspahn and Joseph D. Phillips

Institute for Geophysics
University of Texas at Austin
Austin, Texas 78759

UTIG Technical Report TR-98
Submitted to the Ocean Drilling Program (ODP) and the Joint Oceanographic Institutions/U. S. Science Advisory Committee (JOI/USSAC)
UT Contract 26-3904-25
23 December 1988

Table of Contents

<table>
<thead>
<tr>
<th>General Statement</th>
<th>Program</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programs</td>
<td>README.1ST</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>LISLOAD.FOR</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>LISDUMP.FOR</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>LIS2SEGY.DEF</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>LIS2SEGY.FIN</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>LIS2SEGY.FOR</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>INIT.FOR</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>LLOAD.FOR</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>PLOAD.FOR</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>SEGYPAR.FIN</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>TAPE.FOR</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>TRANSLATE.FOR</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>VAXIBM.MAR</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>DMPGDB.FOR</td>
<td>47</td>
</tr>
</tbody>
</table>

LIS/ACCESS User Agreement Documentation | 51 |
General Statement

The University of Texas Institute for Geophysics (UTIG) under JOI/USSAC sponsorship has undertaken the task of developing computer software programs to translate proprietary Schlumberger LIS format, Vertical Seismic Profile waveform data tapes into standard SEG-Y format, exchange data tapes. The purpose of this software is to enable seismic data processing aboard the Drilling Vessel JOIDES RESOLUTION and at onshore laboratories using non-Schlumberger computers and software. Schlumberger Well Services (Houston) has participated in this effort and has been most cooperative in helping us decode the LIS format field tapes.

Programs

README.1ST

This document describes a suite of mostly Fortran programs running on DEC's VAX computers (under VMS v4.2 or higher) to translate Schlumberger CSU generated LIS format tapes or files into SEGY tapes. These programs will specifically translate LIS-79 WSTA seismometer tool data from the Well Seismic Service (WSS), but may work for other fast data channels as well. The programs are designed be used by persons unfamiliar with LIS and SEG-Y formats in a production environment. M. Wiederspahn was responsible for writing the computer code of the subject programs. J. Phillips initiated the program development and evaluated the results.

There are three programs: LISDUMP, LISLOAD and LIS2SEGY. LISDUMP documents the symbols available from an LIS tape; it is used to determine which should be translated to the SEGY header. This program relies on the Client version of Schlumberger's LIS/Access software, which is considered by them trade secret proprietary information, and must be obtained and used in accordance with a specific Schlumberger user agreement (See attached user agreement/letters between Schlumberger and the University of Texas restricting use of the enclosed translation software).

LISLOAD copies a LIS format tape to a disk file; this capability may be needed if two tape drives are not simultaneously available for LIS2SEGY. LISLOAD is VAX/VMS specific, but does not rely on any other code except standard VMS I/O services.

LIS2SEGY reads LIS tape or disk logical files, translating what is read into one or more SEG-Y output tapes. LIS/Access hides the specifics of the input format from the translation program. The details of the output SEG-Y tape are determined by a text parameter file and may be modified to suit the end use, as needed. The operator may choose to place either one LIS file per SEG-Y output physical reel, which strictly complies with the SEG-Y standard, or multiple SEG-Y tape files onto a physical reel to reduce the number of tapes used, or to place multiple LIS files into one SEG-Y file, as desired.
In general, one SEG-Y tape image will be generated for each LIS logical file, unless the third option is chosen.

Operation is largely automatic. The user defines the logical name LIS_IN as either a tape drive or disk file. The user MOUNT/FOREIGN's the output tape to set its density, and ASSIGN's that name to logical name LIS_OUT. LIS2SEGY is run; the operator is asked to mount the first input reel if the user has not already mounted it /FOREIGN. Should the logical input continue on other reels, the operator is asked for a reel at the end of the each input reels. The translation process may generate multiple output tapes as required by the SEG-Y standard, one output file (or logical tape) per input logical file. LIS2SEGY also makes a log file of LISA information and parameter choices which serves as auxiliary documentation describing the translation.

The resulting SEG-Y tape will usually contain two channels, the source channel and the downhole channel. Each data frame is mapped into one or more traces; each frame/trace is associated with a shot number and a depth. Depths may or may not change during a series of shots. LIS2SEGY makes no attempt to sort out these permutations other than to produce a faithful rendering of its input tape. The user's SEG-Y processing programs must be able to collect common depths for stack or other combinations, based on the CDP and the CDP sequence number. The latter changes to one each time the CDP changes.

LISDUMP Instructions

1) either load the tape on a drive, and type:
   $ASSIGN drive-name LIS_IN
   $MOUNT/FOREIGN LIS_IN
   or
   $ASSIGN diskfilename LIS_IN

2) type:
   $ASSIGN/USER outputfilename SYS$OUTPUT
   $RUN LISDUMP
   $DISMOUNT LIS_IN (if a tape)

3) examine LIS_OUT; look for interesting names. In particular, verify the list of OBJECTS under SETNAME=0, SETTPY=CHAN; compare those on your tape
with those listed in the file LIS2SEGY.DEF. If these agree, you are assured at least of a correct translation. Look at other variables as well; particularly those in the CHAN section. Any of these which your processing requires should be described in LIS2SEGY.DEF as well. There is some discussion in LIS2SEGY.DEF about the meaning of the variables.

LISLOAD Instructions

1) mount the input LIS tape on a tape drive; type:
   $ASSIGN tapedrive LIS_IN
   $MOUNT/FOR LIS_IN
   $ASSIGN diskfilename LIS_OUT
   $RUN LISLOAD
   $DISMOUNT LIS_IN

The disk file can now be used as input for LISDUMP or LIS2SEGY as desired.

LIS2SEGY Instructions

1) read LIS2SEGY.DEF. The chances are very good that it can be used without either change or comprehension. However, if you find that your LIS tape makes many output files, you might want to change from SINGLEFILE to MULTI_FILE or from NORMAL to ALLONE. In general, the latter is the better choice if there are many small files rather than a few large ones. At this writing, I can't really advise you, since Schlumberger seems to change their software frequently, and I don't know what kind of tape you may end up with.

2) either load the tape on a drive, and type:
   $ASSIGN drive-name LIS_IN
   $MOUNT/FOREIGN LIS_IN
   if you have two available drives of the correct densities, or
   $ASSIGN drive-name LIS_IN
   $ASSIGN diskfilename LIS_OUT
   $RUN LISLOAD
   $ASSIGN diskfilename LIS_IN

3) use xxxx=1600 if segy tape is to be read by Masscomp cipher drives. Type:
   $ASSIGN drive-name LIS_OUT
4) type:
$RUN LIS2SEGY

Type an R to rewind the output tape before use

If the program asks you to mount more input or output tapes, do so. Under normal circumstances, input tapes are much less than a whole reel, and so neither requires changing. If you want to terminate the translation before the logical end of the input, you may do so either by responding Q when you are asked if a new output tape is ready, or type a control-C at any time, and manually write an eof at the end of the output tape: set magtape/end lis_out.

$PRINT LIS2SEGY.LOG
$DISMOUNT LIS_OUT
$DISMOUNT LIS_IN

An example of a log file is included in this package. Yours should not look too dissimilar. This particular example has both normal and abnormal printout within it. It is annotated for your edification; see file lis2segy.log_example.
c LISLOAD.FOR

c

t THIS PROGRAM CONTAINS VAX/VMS SPECIFIC CODING!
(c it is also slow as a dog...)
c
22feb88  mark wiederspahn, u of texas institute for geophysics
     8701 Mopac Blvd
     Austin TX 78758
c
to compile and run this program:
c
$fortran lisload
$link lisload
c ...put the input LIS tape on any drive (e.g. ms0:), write ring removed
c
$mount/foreign ms0:
$assign ms0: LIS_IN
$assign target_file LIS_OUT
$run lisload
c
Inputs:
c   input tape is mounted/foreign as logical name 'LIS_IN'
c
Outputs:
c   output file is logical name 'LIS_OUT'.
c
progress reports are output to SYS$OUTPUT
c

---------------------------------------------

program lisload

c
   parameter ( LENREC = 1024 ) ! max rec length

c
   integer sys$assign
   integer sys$alloc
   integer sys$qio

c
   external io$ rewind
   external io$ skiprecord ! tricky way to get linker to
   external io$ readblk ! ...do some of the work for
   external io$ rewino ! ...us
external io$m_nowait
c
integer ss$normal, ss$overun, ss$endoffile, ss$endoftape
external ss$normal
external ss$dataoverun
external ss$endoffile
external ss$endoftape
c
integer*2 ichan
integer*2 iosb(4) ! tape io status
integer nrec ! input record counter
character*80 errtxt ! text of system errors
character*10 device ! name of tape device
character*4 answer ! yes or no
logical more ! T if another tape wanted

c
byte buf(LENREC) ! io buffer
c
c
write(*,1010)
1010 format( t20, 'LISLOAD version 1.0' )
c
c this file receives binary data, so it's not printable anyway
c
c open( unit=1, name='LIS_OUT', form='unformatted',
$ rec=LENREC, recordtype='variable', status='new',
$ initialsize=50, extendsize=50, buffercount=5 )
c
c grab a tape drive
c
c istat = sys$assign( 'LIS_IN', ichan, , )
if( .not. istat ) call lib$stop( %val(istat) )
c
c grab it for exclusive access until program exits
c
c istat = sys$alloc( 'TAPE', ldevice, device, )
C if( .not. istat ) call lib$stop( %val(istat) )
C write(‘*,’ ‘Physical device’, device(1:ldevice),’ allocated.’
c
c repeat tape conversions until operator says no more to do
c start at the beginning of the physical reel
c
c istat = sys$qiow( ,%val(ichan), io$ rewind,
$ iosb,..... )
if (.not. istat) call lib$stop( %val(istat) )
if (.not.iosb(1)) call lib$stop( %val(istat) )
  nfile = 0

for each physical file on this tape, translate until eof, ask user
to proceed to next, unless two eofs means no more files.

  neof = 0
100   continue
     nrec = 0
     nfile = nfile + 1

while not eof or eot: read a block, translate, output

200   call sys$qio(.,%val(ichan), io$_readblk, $   iosb,,,, buf, %val(LENREC),....)

eot should NEVER occur if it is a valid LIS tape

if( iosb(1) .eq. %loc(ss$_endoffile) .or. $   iosb(1) .eq. %loc(ss$_endoftape) ) goto 300
   neof = 0
   nrec = nrec + 1
   neof = 0
if( iosb(1) .ne. %loc(ss$_normal) ) then
   call sys$getmsg( %val(istat), lerrtxt, errtxt, %val(15), )
   write(”,,”) ’error: ’, errtxt(1:lerrtxt)
   write(”,,”) ’at file ’,nfile,’ record’,nrec
else
   write(”,,”) iosb(2)
write(1,1020) (buf(i),i=1,iosb(2))
1020   format( 2000a1 )
       write(1) (buf(i),i=1,iosb(2))
   endif
   goto 200

300   continue.
if unc = %loc(io$_rewindoff) + %loc(io$m_nowait)
call sys$qio(.,%val(ichan), %val(ifunc), $   iosb,,,,,,)

if eot, or eod, do not even ask user whether to continue

   neof = neof+1
if( iosb(1) .eq. %loc(ss$_endoftape) ) neof = 2
    answer = 'n'
c
    if( neof .ne. 2 ) then
        write(*,3010) nfile, nrec
        3010    format( ' end of file ','i4,' records = ',' i6 )
        c400    continue
        c        write(*,4010)
        c4010    format( '/' Do you want to load another file (y/n)? ','$ )
        c        read(*,4020) answer
        c4020    format( a )
c    if( answer .eq. '' ) goto 400
        answer = 'y'
    endif
c
    more = answer(1:1) .eq. 'Y' .or. answer(1:1) .eq. 'y'
    if( more ) goto 100
c
    write(*,4030)
    4030    format( ' end of data or end of tape' // t20, 'LISLOAD done' )
c
    end
C LISDUMP.FOR
C This program reads an LIS file into the GDB and
C dumps its entire contents to the screen.
C
C Include definitions of all program parameters and initialize
C LIS/A.
C
INCLUDE 'LISA:LISDEF.INC'
INCLUDE 'LISA:EXCEPT.INC'
INCLUDE 'LISA:LISEDT.INC'
INTEGER LUR,EC,LCR(SLC),CNT
C
C Initialize LIS/A
C
CALL LISINI(EC)
C
C Start the Log file. LU 0 will result in exception output
C being written to the default file LISA.LOG. 6 writes it
C to the terminal.
C
CALL WRTLOG(6,LMSUCC,EC)
C
C Open input Logical Unit (LU).
C Set the LU Flags to "Rewind", "LIS79" and "Buffered I/O".
C Note that the operator message in LUOPEN is
C ignored since LISA.DAT is a disk file. add MOUNT
C
LCR(FLCFLG) = MLCREW+MLCL79+MLCBUF+MLCMNT
CALL LUOPEN('LIS_IN',LCR,
+      'Please mount LIS input tape',LUR,EC)
IF(IAND(EC,MCC).GT.CCWARN)STOP 'Failed to open Logical Unit'
C
C Open nth Reel
C
WRITE(*,*) 'enter number of the reel to dump'
READ(*,*) nreel
CALL OPNREL(LUR,**,nreel-1,EC)
IF(IAND(EC,MCC).GT.CCWARN)STOP 'Failed to open Logical Reel'
C
C Open nth Tape
C
WRITE(*,*) 'enter number of the tape to dump'
READ(*,*) ntape
CALL OPNTAP(LUR,**,ntape-1,EC)
IF(IAND(EC,MCC).GT.CCWARN)STOP 'Failed to open Logical Tape'

C
C open nth File
C
write('*,*) 'enter number of the file to dump'
read('*,*) nfile
    CALL OPNFIL(LUR,'*',nfile-1,EC)
    IF(IAND(EC,MCC).GT.CCWARN)STOP 'Failed to open Logical File'
    CALL RDFILE(LUR,EC)
    IF(IAND(EC,MCC).GT.CCWARN)
        STOP 'Failed to internalize static information in file'
C
C Call Subroutine DMPGDB to dump the GDB to the screen
C
    CALL DMPGDB
C
C Close the File, Tape, and Reel.
C
    CALL CLSFIL(LUR,EC)
    CALL CLSTAP(LUR,EC)
    CALL CLSREL(LUR,EC)
C
C Close the LU.
C
    CALL LUCLOS(LUR,LCR,EC)
C
C Stop the Log file.
C
    CALL ENDLOG(0,EC)
C
C End the program.
C
    STOP ''
END
c lis2segy.def

c this file provides the minimum necessary and default settings
c to allow lis2segy to run. The second section MUST not contain
c tabs as white space, since fortran is really dumb about these
c things! Tabs are ok before the comments, however.
c See also comments in source file PLOAD.FOR

c c

c the segy output ebcdic header contains all spaces.

c in addition to those things named here, the binary header contains:
c
     c line number     always 1
     c reel number      starts at 1 for each run, increments per reel
     c number of data traces  sum of source+downhole channels
     c sample intervals   from SRSS and SRDS
     c number of samples   from DFCNT call(s)
     c data format code    always 1=ibm floating point

c in addition to those things named here, the trace header contains:
c
     c line sequence number  set to 1 for each run, increments/trace
     c reel sequence number   set to 1 when each "output tape" starts;
c         that is, at the start of each output file
     c trace no. within CDP   set to 1 when the CDP changes, incremented
         by one when CDP stays the same.
     c trace ident. code      set to "seismic data" for all channels

c

c ------------------------ RULES FOR CHANGING THIS FILE

-------------------------------

c for the first line:
c the LIS log level, the higher the number the more the output:
c 1 fatal errors
   2 serious errors
   3 failed operation messages
   4 warning messages
   5 informational messages
   6 successful operation messages
   7 hidden internal LISA messages
   8 control handler messages
the density of the output tape (used only to compute feet used):
800, 1600, 6250

the number of feet to write on the output tape:
1 : 2400 (usually; 2350 is popular)

whether it is permitted to place multiple SEG-Y tape
images on a single output tape:
MULTI_FILE or SINGLEFILE

whether to make all channels (both source and downhole(s))
the same length, as required by some processing systems.
No attempt is made to reconcile sample intervals, which
may be troublesome for such systems also:
ALLSAME or VARIABLE

whether all input data is to map to one output segy tape.
some list tapes seem to have one file per cdp, and this
makes so many output files as to become difficult for
some processing systems to handle.
ALLONE or NORMAL

the second line has the special names as noted above.
each name is 6 characters, with a space between it and the next
c
the third and fourth lines are ignored.
c
subsequent lines are header map instructions.
c
As an example, on the LEG111 WSS tape, these CHAN objects
c existed. These are candidates for having their values loaded
c into SEGY trace headers, except for those marked by *, which
c are always loaded; you will screw things up if you specify
c them here. The units are not guaranteed to remain the same as
c shown here; you should check. There is a very clever indirection
c scheme built into LIS, so that you need not know the exact name
c of the data channels, but can find their name by looking up the
c value of other parameter. Unfortunately, I don't know what these
c rules are, exactly, so I always look for the names themselves.
This will probably work most of the time, except you have to do
c the legwork and fix it if S1 and D1 don't work!
c TOD     time of day, seconds since 1980 (should be julian day),
c some operators enter day of month instead

c CPRE    compressor pressure, bar

c SCAT    stack catalog (?) don't know pattern

c DEPT    depth, meters (always 0 for stack traces)

c DFI     Data File ID; seems to be 0 for raw data or 1 for stack

c SNUM    shot number; of raw shot or stack no. of stack

c CALI    caliper, inches

c BTIM    break time, ms  don't know what this is good for

c TTIM    transit time, ms

c SRSS    * sample "rate" for surface sensor, ms

c SRDS    * sample "rate" for downhole sensor, ms

c TOFS    * time of first sample, surface, ms

c TOFD    * time of first sample, downhole, ms

c SSG1    ?

c SSG2    ? (not in the lis customer subset book)

c SSG3    ? (always seem to be zeroes)

c SSGA    ?

c DSG     downhole seismic gain

c ODFW    offset distance, meter

c OAIZ     offset azimuth, degrees

c ELEV    source elevation, meter

c SDBS    source distance below surface, meter

c STD     day (turns out to be day of month on some tapes)

c STH     hour

c STM     minute

c STS     second

c S1      * source fast channel

c D1      * downhole fast channel

C

C SUGGESTION: do not make any index even; if you do it will
C not fall on i*2 or i*4 boundaries and will be hard for
C most processing systems to accept the value.
C
C specify setnam, settyp, object, label, index, length, and comments
C for each mapping entry. If the name is one of the above, put blanks
C for the label entry. The first value of any parameter is available
C when the binary reel header is created. In my personal experience
C so far, only those above (or similar names appearing in the
C setnam=0 settyp=CHAN category) seem to change during the input file.
C
C index:  <0 put this variable in the binary reel header
C at byte offset abs(n)
C  >0 put this variable in the trace header at
<table>
<thead>
<tr>
<th>setnam</th>
<th>settyp</th>
<th>object</th>
<th>label</th>
<th>index</th>
<th>length</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOOLS</td>
<td>WSTA</td>
<td>LENG</td>
<td>-101</td>
<td>2</td>
<td></td>
<td>wsta tool height (as an example)</td>
</tr>
<tr>
<td>CHAN</td>
<td>SNUM</td>
<td>9</td>
<td>4</td>
<td></td>
<td></td>
<td>shot number -&gt; field rec no.</td>
</tr>
<tr>
<td>CHAN</td>
<td>SNUM</td>
<td>17</td>
<td>4</td>
<td></td>
<td></td>
<td>shot number</td>
</tr>
<tr>
<td>CHAN</td>
<td>DEPT</td>
<td>21</td>
<td>4</td>
<td></td>
<td></td>
<td>depth -&gt; cdp</td>
</tr>
<tr>
<td>CHAN</td>
<td>DFI</td>
<td>35</td>
<td>2</td>
<td></td>
<td></td>
<td>data file number -&gt; data use</td>
</tr>
<tr>
<td>CHAN</td>
<td>STD</td>
<td>159</td>
<td>2</td>
<td></td>
<td></td>
<td>day of year (should be)</td>
</tr>
<tr>
<td>CHAN</td>
<td>STH</td>
<td>161</td>
<td>2</td>
<td></td>
<td></td>
<td>hour</td>
</tr>
<tr>
<td>CHAN</td>
<td>STM</td>
<td>163</td>
<td>2</td>
<td></td>
<td></td>
<td>minute</td>
</tr>
<tr>
<td>CHAN</td>
<td>STS</td>
<td>165</td>
<td>2</td>
<td></td>
<td></td>
<td>sec</td>
</tr>
<tr>
<td>CHAN</td>
<td>TOD</td>
<td>181</td>
<td>4</td>
<td></td>
<td></td>
<td>sec since 1980</td>
</tr>
<tr>
<td>CHAN</td>
<td>CPRE</td>
<td>185</td>
<td>4</td>
<td></td>
<td></td>
<td>compressor pressure, bar</td>
</tr>
<tr>
<td>CHAN</td>
<td>SCAT</td>
<td>189</td>
<td>4</td>
<td></td>
<td></td>
<td>stack catalog?</td>
</tr>
<tr>
<td>CHAN</td>
<td>CALI</td>
<td>193</td>
<td>4</td>
<td></td>
<td></td>
<td>caliper</td>
</tr>
<tr>
<td>CHAN</td>
<td>BTIM</td>
<td>197</td>
<td>4</td>
<td></td>
<td></td>
<td>break time</td>
</tr>
<tr>
<td>CHAN</td>
<td>TTIM</td>
<td>201</td>
<td>4</td>
<td></td>
<td></td>
<td>transit time</td>
</tr>
<tr>
<td>CHAN</td>
<td>SSG1</td>
<td>205</td>
<td>4</td>
<td></td>
<td></td>
<td>?</td>
</tr>
<tr>
<td>CHAN</td>
<td>SSG2</td>
<td>209</td>
<td>4</td>
<td></td>
<td></td>
<td>?</td>
</tr>
<tr>
<td>CHAN</td>
<td>SSG3</td>
<td>213</td>
<td>4</td>
<td></td>
<td></td>
<td>?</td>
</tr>
<tr>
<td>CHAN</td>
<td>SSGA</td>
<td>217</td>
<td>4</td>
<td></td>
<td></td>
<td>?</td>
</tr>
<tr>
<td>CHAN</td>
<td>DSG</td>
<td>221</td>
<td>4</td>
<td></td>
<td></td>
<td>downhole gain?</td>
</tr>
<tr>
<td>CHAN</td>
<td>ODFW</td>
<td>225</td>
<td>4</td>
<td></td>
<td></td>
<td>hole to source offset</td>
</tr>
<tr>
<td>CHAN</td>
<td>OAIZI</td>
<td>229</td>
<td>4</td>
<td></td>
<td></td>
<td>hole to source azimuth</td>
</tr>
<tr>
<td>CHAN</td>
<td>ELEV</td>
<td>233</td>
<td>4</td>
<td></td>
<td></td>
<td>elevation of source</td>
</tr>
<tr>
<td>CHAN</td>
<td>SDBS</td>
<td>237</td>
<td>4</td>
<td></td>
<td></td>
<td>source below surface</td>
</tr>
</tbody>
</table>
lis2segy.fin

include 'lisa:lisdef.inc'
include 'lisa:except.inc'
include 'lisa:lisedt.inc'

include 'segypar.fin'

parameter ( LUNDAT = 1 ) ! data files
parameter ( LUNLOG = 2 ) ! LIS log file

parameter ( MAXPAR = 50 ) ! number of hdr mappings
parameter ( MAXSAMP = 5000 ) ! number of samples/trace
parameter ( MAXCHAN = 2 ) ! number of seismic channels
parameter ( MAXOTHER = MAXPAR ) ! number of non-seis channels

LIS to SEGY header mapping data. The first 4 entities unambiguously find a datum from the current context in the LIS data. The latter two map that datum into the SEGY trace or binary reel header.

character*(SLABEL) setname(MAXPAR)
character*(SLABEL) settype(MAXPAR)
character*(SLABEL) object(MAXPAR)
character*(SLABEL) label(MAXPAR)
integer index[MAXPAR] ! 1:240 = trace hdr start byte
! -1:-400 = bin hdr start byte
integer lendat[MAXPAR] ! 1,2,4 = integer format
! -4 = ibm floating point
integer npar ! number of params loaded

integer errlvl ! LISA logging level
integer density ! output tape density
integer nfeet ! output tape feet
logical mfile ! t if multi-output files
! on a single tape
logical lequal ! t if all traces on output
! tape to be same length
logical onetape ! t if everything maps to
! one logical segy tape

special CHAN objects: we expect that the values will be the same as the variable names, but can be changed by pload.for.
character*(SLABEL) SRSS  ! source channel sample int
character*(SLABEL) SRDS  ! downhole chan sample int
character*(SLABEL) TOFS  ! source time of first samp
character*(SLABEL) TOFD  ! downhole ditto
character*(SLABEL) S1   ! source channel
character*(SLABEL) D1   ! downhole channel

integer  nchan      ! actual number of channels=
integer  nsrc       ! uphole/source channels +
integer  ndown      ! downhole channels
integer  nsamp(MAXCHAN)  ! actual number of samples
integer  mnsamp     ! largest number of samples
real    traces(MAXSAMP,MAXCHAN)  ! fast (seismic) data area
integer  nother      ! number of other channels
real    other(MAXOTHER)   ! other channels data area
integer  iother(MAXOTHER) ! ditto
equivalence (other, iother)
integer  ssamp(MAXCHAN)  ! source sample interval
integer  dsamp(MAXCHAN)  ! downhole sample interval
integer  sdeep(MAXCHAN)  ! source deep water delay
integer  ddeep(MAXCHAN)  ! downhole deep water delay

byte    thdr(LENGTHDR) ! segy trace is comprised of:
integer*2 i2hdr(LENGTHDR/2)  ! trace header of either
integer*4 i4hdr(LENGTHDR/4)  ! bytes,words, or longs
real    data(MAXSAMP)  ! plus a time-series
real    tdata(LENGTHDR/4+MAXSAMP)  ! of (in this case) ibm
equivalence (thdr,i2hdr,i4hdr,tdata) ! floating point data
equivalence (data,tdata(LENGTHDR/4+1))

common  /lis2sgy/
  $  setname, settype, object,label,
$  index, lendat, npar,
$  errlvl, density, nfeet, mfile, lexual, onetape,
$  SRSS, SRDS, TOFS, TOFD, S1, D1,
$  nchan, nsrc, ndown, nsamp, mnsamp, traces,
$  nother, other, ssamp, dsamp, sdeep, ddeep,
$  tdata
c lis2segf.for

c Program to translate Schlumberger LIS WSS Service vertical seismic
profile data tapes to seismic industry standard SEG-Y format tapes.
It allows the user to specify how the reel header and trace headers
are to be created, with minimal assumptions about what should go where.
See the files "lis2segf.def" and "pload.for" for further information.

c inputs:
logical name LIS_IN is the name of an input tape drive or disk file,
previously loaded from tape using program LISLOAD. The
drive need not, but may be already mounted/foreign.

logical name LIS_OUT is the name of an output tape drive. It must
be already mounted/foreign.

disk file LIS2SEGY.DEF contains default parameters as described in
the file PLOAD.F

disk file LIS2SEGY.PAR contains additional optional parameters as
described in PLOAD.F

c usage:
$assign tape_name LIS_IN
$assign tape_name LIS_OUT
$run lis2segf
$print lis2segf.log

outputs:
disk file LIS2SEGY.LOG will contain a summary of this program run.
tape(s) on drive LIS_OUT will have been written in SEG-Y format

recapsulation:
requires Schlumberger proprietary LIS/Access software to
have been previously loaded into directory LISA:

$ @LIS2SEGYBLD.COM

---------------------------------------------
program  lis2segf
include 'lis2segy.fin' ! must be first!
include 'tape.fin' ! output tape control stuff

integer lur ! lisa logical unit number
integer ec ! lisa status
integer lcr(SLC) ! lisa control array
character*10 cgtrel, cgttap, cgtfii ! functions
logical pchange ! t if parameters might have changed
   ! due to interning a transient record
character ans ! y, Y, n, or N

c

open( LUNLOG, file='lis2segy.log', status='new' )
write(*,1010)
write(LUNLOG,1010)
1010 format(17x,'LIS2SEGY version 1.0 August 1988//
   $ ' Translates LIS WSS service WSTA tool raw or stack data into'
   $ ' SEG-Y traces with user specifiable trace header contents.',/
   $ ' Copyright 1988, Mark Wiederspahn /
   $ ' Institute for Geophysics, University of Texas at Austin//
   $ ' This program makes use of Schlumberger proprietary',/
   $ ' object code and thus must only be used in accordance with'
   $ ' their stated restrictions and policies. It is to be used'
   $ ' by the Ocean Drilling Program aboard JOIDES Resolution,/'
   $ ' and at Texas A&M as needed to support ship operations.'// )

c
call init( ier ) ! load parameter arrays
call tinit( ier ) ! grab output tape drive
if( ier .gt. 0 ) then
call lisini( ec ) ! init LIS access software
call wrtlog( LUNLOG, errlvl, ec )
100 lcr(FLCFLG) = MLCBUF + MLCL79 + MLCMNT ! automount if not up
call luopen( 'LIS_IN', lcr, 'Mount LIS WSS INPUT tape',
   $ lur, ec )
if( iand(ec,MCC) .gt. CCWARN ) stop 'LUOPEN failed'
c
c for every file on every logical reel, make an output file
c
200 call opnrel( lur,**,0,ec )
   if( iand(ec,MCC) .gt. CCWARN ) goto 900
   write(LUNLOG,* ' start reel **, cgtrel(ec),'''
write(LUNLOG,*) ' start reel "", cgtrcl(ec),""
write(*,*) ' start reel "", cgtrcl(ec),""
call dishr( LUNLOG,ec )

300  call opntap( lur,"",0,ec )
if( iand(ec,MCC) .gt. CCWARN ) goto 800
  write(LUNLOG,*) ' start tape "", cgttap(ec),"
write(*,*) ' start tape "", cgttap(ec),"
call dishr( LUNLOG,ec )

c
400  call opnfil( lur,"",0,ec )
if( iand(ec,MCC) .gt. CCWARN ) goto 700
  write(LUNLOG,*) ' start file "", cgffil(ec),"
write(*,*) ' start file "", cgffil(ec),"
call dishr( LUNLOG,ec )
call rdfile( lur,ec )
if( iand(ec,MCC) .le. CCWARN ) then
  call load( lur,ec )  ! load channels data
if( iand(ec,MCC) .le. CCWARN ) then

once tape mounted, "reelup" is true, and remains so until user terminates
by saying "quit" rather than continue when asked for new output tape;
thus the tests for reelup at the bottom of each loop.
If we are aborting out of here, then there is no point keeping the
c correct positions by sequential calls to the flavors of lis closes (?)
(these are very time consuming because it seems to READ rather than skip).
if we won't break segy file at each lis file, then only at the beginning
do we "open an output".

ier = 1
if( onetape ) then
  if( .not. reelup ) call opnout( ier )
else
  call opnout( ier )
endif
if( ier .ge. 0 ) then  ! tape is up and open
  pchange = .true.  ! get gdb params first t
ime

nframe = 0
ntrans = 0  ! purely advisory

500  continue
  call lisnxt( lur,0,ec )
if( iand(ec,MCC) .le. CCWARN ) then
  nframe = nframe + 1
call translate( pchange,ier )
pchange = .false.
else
    if( ec .eq. EFMTR ) then
      ntrans = ntrans + 1
      pchange = .true.
call rdgdb( lur,ec ) ! intern transients in g
  endif
else
  ...undocumented status from lisnxt...
  seems to be building fast channel data?
iand(ec,MCC) = 7, control
  endif
endif
if( reelup .and. (ec .ne. EFMNF) ) goto 500 ! not ye
 t EOF
endif
endif
endif
write(LUNLOG,*) ' end file "", cgtfil(ec),""
write(*,*) ' end file "", cgtfil(ec),"
if( reelup ) call clsfil( lur,ec ) ! close input logical file
call clrfil( ec ) ! zap gdb info for this file
if( nframe .gt. 0 .and. .not. onetape )
  call clsout( ier ) ! close output file
write(LUNLOG,5010) nframe
write(*,5010) nframe
5010    format( 9x,'data frames translated:', i7 )
write(LUNLOG,5020) ntrans
write(*,5020) ntrans
5020    format( 9x,'transient records seen:', i7 / )
if( reelup ) goto 400 ! if still have output tape

500    continue
write(LUNLOG,*) ' end tape "", cgttap(ec),""
write(*,*) ' end tape "", cgttap(ec),"
if( reelup ) call clstap( lur,ec ) ! close input logical tape
call clrtap( ec ) ! zap gdb info for tape
if ( reelup ) goto 300

800 continue
write(LUNLOG,*) ' end reel "", cgtrel(ec),""
write(*,*) ' end reel "", cgtrel(ec),""
if ( reelup ) call clsrel( lur,ec )       ! close input logical reel
call clrrel( ec )                        ! zap gdb info for reel
if ( reelup ) goto 200

900 continue
if ( onetape ) call clsout( ier )       ! if not closed at end of file
ans = 'N'
if ( .not. reelup ) then
   write(*,9020)
9020 format( 'Continue with another input tape?';$ )
   read(*,9030) ans
9030 format( a )
endif
if ( ans .eq. 'Y' .or. ans .eq. 'y' ) then
   ans = 'Y'
call luclos( lur,lcr,ec )               ! close input
else
   lcr(FLCFLG) = lcr(FLCFLG) + MLCDMT + MLCUNL
call luclos( lur,lcr,ec )               ! close input
call unlout( ier )                     ! unload output tape
endif
call clrlu( ec )                        ! clear this lu from the gdb
if ( ans .eq. 'Y' ) goto 100            ! get another tape
endif
cend

c
c
init.for

c initialize whatever needs to be done once:
load required parameter mappings
load optional parameter mappings

subroutine init( ier )
integer ier ! output: >=0 if ok, <0 if error

include 'lis2segy.fin'

nchan = 0
nsrc = 0
ndown = 0

npair = 0
open( LUNDAT, file='lis2segy.def',status='old',iostat=ios )
if( ios .ne. 0 ) then
  stop 'missing required file LIS2SEGY.DEF'
endif

call pload( LUNDAT,ier ) ! load required params

c KISS - keep it simple, stupid

c if( ier .gt. 0 ) then
c  close( LUNDAT )
copen( LUNDAT, file='lis2segy.par',status='old',iostat=ios )
c if( ios .ne. 0 ) then
c  call pload( LUNDAT,ier )
c  close( LUNDAT )
c endif
c endif
c
return
end
c lload.for

c load internal transformation info about the LIS tape

c subroutine llload( lur, ec )
integer lur ! in: lu for lis
integer ec ! out: status

c include 'lis2segy.fin'

c the following settype CHAN objects are "known" to us,
c and are specially treated here: S1 D1 SRSS SRDS TOFS TOFD

c It is unfortunate that they did not treat the source channel
c as a kind of downhole channel, since that would regularize
c this processing, and we would not have to identify then
c as exceptional classes.
c
c call dfcrrv( traces, 1, 0, S1, 0, 0, nsr, "", ec )
if( iand(ec, MCC) .gt. CCWARN ) write('*, *) 'err in ', S1
if( iand(ec, MCC) .gt. CCWARN ) write(LUNLOG,*) 'err in ', S1
write(LUNLOG, 1010) nsr
write('*,1010) nsr
1010 format( t10, 'found ', i2, ' source channels.' )


call dfcrrv( traces, 1, 0, D1, 0, 0, ndow, "", ec )
if( iand(ec, MCC) .gt. CCWARN ) write('*, *) 'err in ', D1
if( iand(ec, MCC) .gt. CCWARN ) write(LUNLOG,*) 'err in ', D1
write(LUNLOG, 1020) ndow
write('*,1020) ndow
1020 format( t10, 'found ', i2, ' downhole channels.' )
nchan = nsr + ndow
if( nchan .eq. 0 ) goto 999 ! error abort

c if( nchan .gt. MAXCHAN ) then
write('*,*) 'increase MAXCHAN to ', nchan, ' in file', $
   'lis2segy.fin and recompile.'
stop
endif

c identify the traces and counts to be automatically loaded via call to lisnxst
mnsamp = 0 ! max nsamp
do 100 i=1, nsrccall dfclvr( traces(1,i),MAXSAMP,0,
$     S1, i,0,junk,",",ec )
call dfcnt( nsamp(i),ec )
if( iand(ec,MCC) .gt. CCWARN ) write(LUNLOG,* ) 'err in ',S1
100  continue
c
do 200 i=1, ndown
call dfclvr( traces(1,i+nsrcc),MAXSAMP,0,
$     D1, i,0,junk,",",ec )
call dfcnt( nsamp(i+nsrcc),ec )
if( iand(ec,MCC) .gt. CCWARN ) write(LUNLOG,* ) 'err in ',D1
200  continue
c other special cases; units to SEGY values
c
call dfclvr( ssamp,MAXCHAN,0, SRSS, 1,0,j,'US',ec )
if( iand(ec,MCC) .gt. CCWARN ) write(LUNLOG,* ) 'err in ',SRSS
call dfclvr( dsamp,MAXCHAN,0, SRDS, 1,0,j,'US',ec )
if( iand(ec,MCC) .gt. CCWARN ) write(LUNLOG,* ) 'err in ',SRDS
call dfclvr( sdeep,MAXCHAN,0, TOFS, 1,0,j,'MS',ec )
if( iand(ec,MCC) .gt. CCWARN ) write(LUNLOG,* ) 'err in ',TOFS
call dfclvr( ddeep,MAXCHAN,0, TOFD, 1,0,j,'MS',ec )
if( iand(ec,MCC) .gt. CCWARN ) write(LUNLOG,* ) 'err in ',TOFD
c c set up all other channels
c for each header value which is of settype CHAN, try to
c make it known to LIS, and hope that a) there is only one
c channel with that name, and b) it is a single valued channel.
c
other = 1
do 300 i=1, npar
if( (index(i) .gt. 0) .and. (settype(i) .eq. 'CHAN') ) then
call dfclvr( other(i),1,0,object(i),0,0,n,"",ec )
if( iand(ec,MCC) .gt. CCWARN )
$     write(LUNLOG,* ) 'err in ',object(i)
d     write(LUNLOG,* ) 'CHAN obj ',object(i),' has ",n," vals'
if( n .ne. 1 ) then
     write(LUNLOG,* ) ' warning - channel ",label(i),
$     ' has ",n," instances/frame.'
     write(LUNLOG,* ) ' warning - channel ",label(i),
$     ' has ",n," instances/frame.'
endif
call dfcivr( other(i),1,0,object(i),1,0,n,"",ec )
if( iand(ec,MCC) .gt. CCWARN )
$  write(LUNLOG,*') 'err in ',object(i)
c c it would be nice to check these out, but we can't call dfcivr
 c if lisnxt has been called, and we can't use these routines
 c until lisnxt or luread has been called. there is not description
 c of the difference between lisnxt and luread, but maybe lisnxt
 c uses luread, then does channel mapping... Anyway, there are
 c more important things to worry about.
c c    call fmgnch( lur,0,object(i),1,islot,ec )
c c    call fmgnch( lur,0,islot,n,ec )
   cd    write(LUNLOG,*') 'CHAN obj ',object(i),' has ',n,' @slot ',islot
   c    if( n .ne. 1 ) then
   c        write(LUNLOG,*') ' warning - channel ',label(i),
   c$         ' has ',n,' values/instance.'
   c    write(LUNLOG,*') ' warning - channel ',label(i),
   c$         ' has ',n,' values/instance.'
   c    endif
   c      nother = nother + 1
   endif
300  continue
 c
 c      nother = nother - 1
 c
999  if( nchan .le. 0 ) then
   write(LUNLOG,*') 'no data channels to translate in this file!'
   ec = ECFERR ! oops
   endif
 c
 c      return
 end
 c
c

c pload.for

c load parameters from open ascii file on lun, skipping

c comments at the beginning of the file until a blank

c line is seen. The next line is parameters followed by two ignored

c lines and then multiple lines of segy header mapping parameters.

c

c These values may be defaulted by spaces, BUT the commas must

c be retained for place keeping.

c

c the LIS log level, the higher the number the more the output:

c 1 fatal errors

c 2 serious errors

c 3 failed operation messages

c 4 warning messages

c 5 informational messages

c 6 successful operation messages

c 7 hidden internal LISA messages

c 8 control handler messages

c

c the density of the output tape:

c 800, 1600, 6250

c

c the number of feet to write on the output tape:

c 1: 2400 (usually; 2350 is popular)

c

c whether it is permitted to place mutiple SEG-Y tape

c images on a single output tape:

c MULTI_FILE or SINGLEFILE

c

c whether to make all channels (both source and downhole(s))
c the same length, as required by some processing systems.
c No attempt is made to reconcile sample intervals, which

c may be troublesome for such systems also:

c ALLSAME or VARIABLE

c

c whether all input data is to map to one output segy tape.
c some lis tapes seems to have one file per cpd, and this

c rapidly makes so many output files as to become difficult

c for some processing systems to handle.

c ALLONE or NORMAL

c

c the next line is 6 special CHAN object names:
c source sample rate, downhole sample rate, time of first source sample,
c time of first downhole sample, sourcechannel, downholechannel
c
c for example: see lis2segy.def
c
c subroutine pload( lun,ier )
  integer lun    ! in: logical unit
  integer ier    ! out: status >=0 ok, <0 error
  include 'lis2segy.fin'
c
  logical done
  character*80 line
  character*10 files
  character*8 lengths
  character*6 tapes
  character*(SLABEL) names(6)
c
  ier = -1
  100 read( lun,'(a)',end=900 ) line
    if( line .ne. '' ) goto 100
  c decode the 'various' parameters
  c
    read( lun,1020 ) j, k, l, files, lengths, tapes
  1020 format( 3i10,x,a,x,a,x,a )
    if( i .ne. 0 ) errvl = j
    if( j .ne. 0 ) density = k
    if( k .ne. 0 ) nfeet = l
    if( files .eq. 'MULTI_FILE' ) then
      mfile = .true.
      elseif( files .eq. 'SINGLEFILE' ) then
        mfile = .false.
      endif
    if( lengths .eq. 'ALLSAME' ) then
      lequal = .true.
      elseif( lengths .eq. 'VARIABLE' ) then
        lequal = .false.
      endif
    if( tapes .eq. 'ALLONE' ) then
onetape = .true.
elseif( tapes .eq. 'NORMAL' ) then
   onetape = .false.
endif
read( lun,1030 ) names
1030    format( a,x,a,x,a,x,a,x,a,a,a)
if( names(1) .ne. ' ' ) SRSS = names(1)
if( names(2) .ne. ' ' ) SRDS = names(2)
if( names(3) .ne. ' ' ) TOFS = names(3)
if( names(4) .ne. ' ' ) TOFD = names(4)
if( names(5) .ne. ' ' ) S1  = names(5)
if( names(6) .ne. ' ' ) D1  = names(6)

read( lun,1030 ) ! skip a line
read( lun,1030 ) ! skip a line

c
c while not eof or too many parameters, load 'em
c
   i = npar+1
   if( i .gt. MAXPAR ) stop 'increase MAXPAR in lis2segf.fin'
c
200    read( lun,1040,end=300 ) setname(i),settype(i),object(i),
   $ label(i), index(i), lendat(i)
1040    format( a,x,a,x,a,x,a , 2i10 )
d    write(*,* ) setname(i),settype(i),object(i),
d    $ label(i), index(i), lendat(i)
   if( setname(i) .eq. ' ' ) goto 200 ! ignore blank lines
   if( index(i)-abs(lendat(i)) .lt. -LENBHDR-1 .or.
   $ index(i)+abs(lendat(i)) .gt. LENTHDR+1 )
   $ write(*,* ) 'index error for ',settype(i),object(i)

   i = i + 1
   if( i .le. MAXPAR ) goto 200
c
   stop 'increase MAXPAR in lis2segf.fin'
c
300    continue
   npar = i-1
c
900    continue
c
   return
c
end
segpar.fin  (previously was segyparams.fin)

byte lengths of ebcdic header, binary reel header, trace header, number of bytes per sample

parameter  ( LENEHDR  = 3200 )
parameter  ( LENBHDR  = 400  )
parameter  ( LENTHDR  = 240  )
parameter  ( NBSAMPLE = 4  )

ALL INDICIES ARE PREDICATED ON BEING ADDED TO THE BASE INDEX
(NOT ADDRESS) OF THE ARRAY. For example, lbuf(l4JOBID+1)

indices into the binary reel header, in i*2 or i*4 units

parameter  ( i4jobid  = (1-1)/4 )
parameter  ( i4line   = (5-1)/4 )
parameter  ( i4reel   = (9-1)/4 )
parameter  ( i2ndtrc = (13-1)/2 )
parameter  ( i2rsint  = (17-1)/2 )
parameter  ( i2fsint  = (19-1)/2 )
parameter  ( i2maxsamp=(21-1)/2 )
parameter  ( i2fsamp = (23-1)/2 )
parameter  ( i2dtype = (25-1)/2 )
parameter  ( i2fold  = (27-1)/2 )
parameter  ( i2sort  = (29-1)/2 )
parameter  ( i2units = (35-1)/2 )

these are essential indicies into trace headers located in any buffer, but most particularly in common /fcore/. Both i*2 and i*4 are needed.

parameter  ( l4LSEQNO = (1-1)/4 )
parameter  ( l4RSEQNO = (5-1)/4 )
parameter  ( l4FFID   = (9-1)/4 )
parameter  ( l4CHAN  = (13-1)/4 )
parameter  ( l4SHOT  = (17-1)/4 )
parameter  ( l4CDPNO = (21-1)/4 )
parameter  ( l4CDPSEQ = (25-1)/4 )
parameter  ( l2TRCID = (29-1)/2 )
parameter  ( l2DATAUSE= (35-1)/2 )
parameter  ( l2SRDIST = (37-1)/2 )
parameter ( i2COORDS = (89-1)/2 )
parameter ( i2LAGA = (105-1)/2 )
parameter ( i2DELAY = (109-1)/2 )
parameter ( i2NSAMPLE= (115-1)/2 )
parameter ( i2SINT = (117-1)/2 )
parameter ( i2INSTG = (121-1)/2 )
parameter ( i2INITG = (123-1)/2 )
parameter ( i2YEAR = (157-1)/2 )
parameter ( i2DAY = (159-1)/2 )
parameter ( i2HOUR = (161-1)/2 )
parameter ( i2MIN = (163-1)/2 )
parameter ( i2SEC = (165-1)/2 )
parameter ( i2TBASE = (167-1)/2 )
parameter ( i2TRCWT = (169-1)/2 )

c
c tape.for vms/segy tape interface for lis2segy
c
c-------------------------------------------------------------

.
c

subroutine tinit( ier )

include 'tape.fin'
character*64 device
c
c
c grab an output tape drive; mark reel as not yet mounted.
c
   istat = sys$assign( 'LIS_OUT', ichan, , )
   if( .not. istat ) call lib$stop( %val(istat) )
c
c grab it for exclusive access until program exits

c
   istat = sys$alloc( 'LIS_OUT', ldevice, device, )
   if( .not. istat ) call lib$stop( %val(istat) )
   C write(*,*) 'Physical device ', device(1:ldevice),' allocated.'
c
   reelup = .false.
   feet = 0.0
   if( density .le. 3200 ) then
      gap = 0.6
   else
      gap = 0.3
endif
lineseq = 0   ! pre-increment it
reelno = 0    ! pre-increment it

   c
   ier = 1
   c
   return
   end

---------------------------------------------------------------

subroutine opnout( ier )
   c
   include 'tape.fin'
   include 'lis2segfin'
   c
   character   ans   ! q = quit, anything else is ok
   c
   c
   c if there is a tape there, and we do not allow multiple files/tape, zap it
   c
   if( .not. mfile ) call unlout( ier )
   c
   c if we have no tape now, get one... otherwise leave tape where it is...
   c
   ier = +1
   if( .not. reelup ) then
      write(*,*)
      write(*,*) 'enter "Q" or "q" if you want to quit this run,'
      write(*,*) 'if there were no channels found, just type <RETURN>.'
      write(*,*)
      write(*,*) 'Please load output tape; if positioned ok now'
      write(*,*) 'and ready now type <RETURN>,'
      write(*,*) 'type "R" or "r" to rewind first'
      read(*,'(a1)') ans
      if( ans .eq. 'Q' .and. ans .eq. 'q' ) then
         if( ans .eq. 'R' .or. ans .eq. 'r' ) then
            istat = sys$qiow(,%val(ichan),io$s rewind,$
            iosb,,,)
            if( .not. istat ) call lib$stop( %val(istat) )
            if( iosb(1) .ne. %loc(ss$_normal) ) then
               istat = iosb(1)
               call sys$getmsg( %val(istat),lerrtxt,errtxt,%val(15), )
write(*,*) 'rewind error: ', errtxt(1,lerrtxt)
write(*,*) 'verify the tape is loaded and online'
else
reelup = .true.
endif
else
reelup = .true. ! too lazy to test online status
iosb(1) = %loc(ss$$_normal)
endif
else
iosb(1) = %loc(ss$$_normal)
ier = -1
endif
if( iosb(1) .ne. %loc(ss$$_normal) ) goto 100
endif

! write the ebcidic and binary headers
!
! this section has been moved to wrtout
! because we don't have enough info about
! the lis data before the first frame is read
!
bot = .true.
!
return
end

------------------------------------------------------------------------

subroutine wrtout( first, lbuf, nbyte, ier )
logical first ! input: t if this trace is the first of a shot
integer lbuf(*) ! input: array to be output
integer nbyte ! input: number of bytes to output
integer ier ! output: <=0 error, >0 ok
!
include 'lis2segy.fin'
include 'tapesegy.fin'
!
save icdpc, icdpseq
!
if( .not.reelup ) then
   write(*,*) 'internal logic error - wrtout called with no reel up!'
endif
c

 c if we are a beginning of logical output tape, write headers
 c
 c ebcdic header is:
 c all blanks
 c binary header is:
 c line number = 1
 c reel number = from start of job = 1
 c number of traces (reel=downhole, field=source)
 c sample intervals
 c sample format code
 c sort code = 1 as shot
 c plus user options from lis2segy.def and lis2segy.par
 c for now...
 c

 if( bot ) then
   icdp = -1
   bot = .false.
   do 100 i=1, LENEHDR
     ebchdr(i:i) = ' ' !space
 100   continue
   call a2e( ebchdr )
  istat = sys$qiow( ,%val(ichan), io$ _writevblk,
 $   io$sb,,,%ref(ebchdr), %val(LENEHDR),**, )
 if( .not. istat ) call lib$stop( %val(istat) )
 if( iosb(1) .ne. %loc(ss$ _normal) ) then
   istat = iosb(1)
   call sys$getmsg( %val(istat),lerrtxt,erretxt,%val(15), )
   write(*,*),'write ebc hdrt: ',erretxt(1:lerrtxt)
 endif
 c

 do 200 i=1, LENBHDR/4
   i4bhdr(i) = 0
 200   continue
 c

   reelno = reelno + 1
   write(LUNLOG,1010) reelno
   write(*,1010) reelno
 1010 format( /x,'-------- Start Output Tape ' ,i3,' --------'//
 $   x,' sequence shot cdp' / )
   i4bhdr(i4LINE+1) = i4swap( 1 )
   i4bhdr(i4REEL+1) = i4swap( reelno )
   i2bhdr(i2NDTRC+1) = i2swap( nchan )
   i2bhdr(i2RSINT+1) = i2swap( dsamp )
i2bhdr(l2FSINT+1) = i2swap( ssamp )
i2bhdr(l2MAXSAMP+1) = i2swap( nsamp(nsrc+1) )
i2bhdr(l2FSAMP+1) = i2swap( nsamp(1) )
i2bhdr(l2DTYPE+1) = i2swap( 1 )
i2bhdr(l2SORT+1) = i2swap( 1 )

c
    call iniset( ec )    ! position to beginning of tape
do 300 j=1, npar
    l = lendat(j)
    if( index(j) .lt. 0 ) then
        write(*,*), 'position GDB to ',
        ! label(j), object(j), settype(j)
        call gdbfind( label(j), object(j),
        settype(j), setname(j),
        '***',***,'***',0,ec )
        if( l .gt. 0 ) then
            ival = igtvl( label(j), ***, ec )
            write(*,*), 'got ',label(j),ival
            if( l .eq. 4 ) then
                ival = i4swap( ival )
            elseif( l .eq. 2 ) then
                ival = i2swap( ival )
            endif
        else
            rval = rgtvl( label(j), ***, ec )
            call vaxibm( rval, ival, 1 )
        endif
        iother(j) = ival        ! save bit pattern
    endif
    call moveb( ival, bhdr(-index(j)), abs(l) )
    300 continue

c
    istat = sys$qiow( ,%val(ichan), io$writeblk,
        $      iosb,,, bhdr, %val(LENBHDR),,,, )
    if( .not. istat ) call lib$stop( %val(istat) )
    if( iosb(1) .ne. %loc(ss$normal) ) then
        istat = iosb(1)
        call sys$getmsg( %val(istat), lerrtxt, errtxt, %val(15), )
        write(*,*), 'write bin hdr: ', errtxt(1:lerrtxt)
    endif

c
    feet = feet + ((LENEHDR+LENBHDR)/density + (gap*2)) / 12.0
    reelseq = 0        ! pre-increment
endif
c if it is time for a change, keep shots together
c
ier = 1
d write(*,*),'wrtout:*,first,feet,reelseq,lineseq
if( first .and. (feet .ge. nfeet) ) then
call cslout( ier )
call unlout( ier )
call opnout( ier )
endif
c
c set the tape record sequence numbers
c
reelseq = reelseq + 1
lineseq = lineseq + 1
libuf(l4LSEQNO+1) = i4swap( lineseq )
libuf(l4RSEQNO+1) = i4swap( reelseq )
c
c manage the cdp sequence number: when the CDP number changes,
c set the sequence number to 1; for each "shot", increment the
c CDP sequence number if the CDP number has not changed.
c If you don't want this feature, don't change the CDP!
c
if( libuf(l4CDPNO+1) .ne. icdp ) then
   icdp = libuf(l4CDPNO+1)
icdpseq = 0
endif
c
if( first ) then ! check once/shot
   if( icdpseq .eq. 0 ) then ! change in cdp
      write(LUNLOG,1020) i4swap(libuf(l4RSEQNO+1)),
$   i4swap(libuf(l4SHOT+1)),
$   i4swap(libuf(l4CDPNO+1))
      write(*,1020) i4swap(libuf(l4RSEQNO+1)),
$   i4swap(libuf(l4SHOT+1)),
$   i4swap(libuf(l4CDPNO+1))
1020     format( x,3i10 )
   endif
   icdpseq = icdpseq + 1
   endif
libuf(l4CDPSEQ+1) = i4swap( icdpseq )
c
c
c write a wrecord; if eot, try again on a new tape; nice shot boundaries
800  continue
  if( ier .gt. 0 ) then
    istat = sys$qiow( ,%val(ichan), io$_writevblk, $  iosb,,, lbuf, %val(nbyte),,,, )
    if( .not. istat ) call lib$stop( %val(istat) )
    ier = 1
    if( iosb(1) .eq. %loc(ss$_endoftape)) then
      call clsout( ier )
      call unlout( ier )
      call opnout( ier ) ! -1 if user quits
      reelseq = reelseq + 1
      lineseq = lineseq + 1
      lbuf(l4LSEQNO+1) = i4swap( lineseq )
      lbuf(l4RSEQNO+1) = i4swap( reelseq )
    else
      if( iosb(1) .ne. %loc(ss$_normal) ) then
        istat = iosb(1)
        call sys$getmsg( %val(istat),lerrtxt,errtxt,%val(15), )
        write(LUNLOG,1030) lbuf(l4RSEQNO+1),errtxt(1:lerrtxt)
        write(*,1030) lbuf(l4RSEQNO+1),errtxt(1:lerrtxt)
      1030 format( 6x,'output error at sequence ',i10,x,a )
    endif
  endif
  endif
  if( (ier .gt. 0) .and.
      $  (iosb(1) .eq. %loc(ss$_endoftape)) ) goto 800

  feet = feet + (nbyte/density + gap) / 12.0
  c
  return
  end

-----------------------------------------------------------------------------------

  subroutine clsout( ier )
  integer ier ! output: <=0 error, >0 ok
  c
  include  'lis2segy.fin'
  include  'tape.fin'
  c
  c
c if an output reel is up now, write EOD (two end of file marks)
c and leave the tape positioned between them, in case there is
c more output on this tape.
c
neof = 2
if( reelup ) then
  write(LUNLOG,1010) reeln0
  write(*,1010) reeln0
1010   format( x,'-------------- End Output Tape ',i3,' --------------' )
do 100 i=1, neof
   istat = sys$qiow( ,%val(ichan), io$_writeof,
$     iosb,,,,,, )
   if( .not. istat ) call lib$stop( %val(istat) )
   if( iosb(1) ,ne. %loc(ss$_normal) ) then
      istat = iosb(1)
      call sys$getmsg( %val(istat),lerrtxt,errtxt,%val(15), )
      write(*,*),'write eof: ',errtxt(1:lerrtxt)
   endif
100   continue
   do 200 i=1, neof-1
      istat = sys$qiow( ,%val(ichan), io$_skiprecord,
$     iosb,,,, ,%val(-1),,, )
      if( .not. istat ) call lib$stop( %val(istat) )
200   continue
   endif
c
feet = feet + neof*(gap+3.0)/12.0 ! hardly worth keeping track of
c
ier = 1
c
c
return
end

-------------------------------------------------------------------

subroutine unlout( ier )
integer  ier   ! output: <=0 error, >0 ok
c
include 'tape.fin'
c
c
if an output reel is up now, unload it
c
if ( reelup ) then
  ifunc = %loc(io$ _rewindoff) + %loc(io$m_nowait)
  istat = sys$qiow( ,%val(ichan), %val(ifunc),
    $iosb,......,)
  reelup = .false.
  feet = 0.0
endif

c
  ier = 1

c
  return
end

---

function swaps bytes in an i2 integer for ibm to dec tape byte order fixup
function may have to compile with nooverflowdetect to keep from blowing up

c
  integer function i2swap( arg )
  byte arg(1)
c
c
  i1 = arg(1)
i2 = arg(2)
if ( i2 .lt. 0 ) i2 = i2 + 256
i2swap = i1*256 + i2
c
  return
end

---

function swaps bytes in an i4 integer for ibm to dec tape byte order fixup
function may have to compile with nooverflowdetect to keep from blowing up

c
  integer function i4swap( arg )
  byte arg(1)
c
c
  i1 = arg(1)
i2 = arg(2)
i3 = arg(3)
i4 = arg(4)
if( i2 .lt. 0 ) i2 = i2 + 256
if( i3 .lt. 0 ) i3 = i3 + 256
if( i4 .lt. 0 ) i4 = i4 + 256

\texttt{i4swap = i1 \ast 256*256*256 +
\$ i2 \ast 256*256 +
\$ i3 \ast 256 +
\$ i4}

\texttt{return}
\texttt{end}

\texttt{
---
}

\texttt{---
}

\texttt{subroutine moveb( in,out,n )
byte in(*)
byte out(*)
integer n
}

\texttt{do 100 i=1,n
out(i) = in(i)
100 continue}

\texttt{return}
\texttt{end}

\texttt{
---
}

\texttt{---
}

\texttt{translate an ascii character string to an ebcidic character string}

\texttt{subroutine a2e( buf )
character*(*) buf
}

\texttt{call lib$tra_asc_ebc( buf, buf )
}

\texttt{return}
end

c

c
c translate.for
c
subroutine translate( pchange, ier )
logical pchange ! in: t if gdb data may have changed
integer ier ! out: conversion status

c
include 'lis2segv.fin'
c
logical first
integer ec
c
c
do this section once only, after the first input from lis
c
if( mnsamp .eq. 0 ) then
do 10 i=1, nchan
   if( nsamp(i) .gt. mnsamp ) mnsamp = nsamp(i)
10 continue
do 20 i=1, THDRLEN
   thdr(i) = 0
20 continue
endif
c
build the header common to the entire shot,
c (include values from the gdb if they could have changed),
c convert all source channels,
c convert all downhole channels

c
first = .true. ! 1st trace of shot
call hdr( pchange ) ! do both gdb and normal channels
n = 1 ! channel number
do 200 i=1, nsrc
   len = nsamp(i)
call vaxibm( traces(1,i), data, len )
   if( lequal .and. (len .ne. mnsamp) ) then
      do 100 j=len+1, mnsamp
         data(j) = 0.0
100 continue
   len = mnsamp
endif
i2hdr(I2SINT+1) = i2swap( ssamp(i) ) ! special channels
i2hdr(I2DELAY+1) = i2swap( sdeep(i) )
i4hdr(I4CHAN+1) = i4swap( n )
i2hdr(I2SAMPLE+1) = i2swap( len )
i2hdr(l2TRCID+1) = i2swap(1) ! seismic data
d  write(*,*) (traces(j,i),j=1,6),len,ssamp(i),sdeep(i),n
call wrtout( first, tdata, len*4+LENTHDR, ier )
  first = .false.
  n = n + 1
200  continue
c
do 400 i=1, ndown
  j = nsrc + i
  len = nsamp(j)
call vaxibm( traces(1,j), data, len )
  if( lequal .and. (len .ne. mnsamp) ) then
    do 300 j=len+1, mnsamp
      data(k) = 0.0
  300    continue
  len = mnsamp
endif
i2hdr(l2SINT+1) = i2swap( dsamp(i) ) ! special channels
i2hdr(l2DELAY+1) = i2swap( ddeep(i) )
i4hdr(l4CHAN+1) = i4swap( n )
i2hdr(l2NSAMPLE+1) = i2swap( len )
i2hdr(l2TRCID+1) = i2swap( 1 ) ! seismic data
d  write(*,*) (traces(k,j),k=1,6),len,dsamp(j),ddeep(j),n
call wrtout( first, tdata, len*4+LENTHDR, ier )
  first = .false.
  n = n + 1
400  continue
c  return
cend
c
C-------------------------------------------------------------
-
c subroutine hdr( pchange )
  logical pchange ! in: t if gdb may have changed
c
  include 'lis2segy.fin'
  integer ec
c
c for each parameter: if it is in the gdb, get its value,
c (actually get it from the gdb only if any transient
c records have been interned, otherwise, copy its value
c saved AS A SEGY HEADER BIT PATTERN in array iother(i))
c otherwise it was a channel, get its current value from
c the frame (done automatically by LISNXT since we DFCRVR'ed
c all the channel params in LLOAD, so they exist AS REAL
c HOST FLOAT variables in array other(i)),
c and plug it into the segy trace header with a byte move.
c
c called once per shot (aka frame)

c
if( pchange ) call iniset( ec )  ! point to current file, 1st set

c
do 100 j=1, npar
   if( index(j) .gt. 0 ) then  ! must be trace header, not reel hdr
      l = lendat(j)
      if( label(j) .ne. ' ' ) then  ! it is a gdb datum
         tval = -1  ! don't know host form of
         ival = iother(j)  ! ...segy previous value
         if( pchange ) then  ! ...in case of no change
            write(".*",'position GDB to ',label(j),object(j),settype(j)
call gdbfdn( label(j), object(j),
         "$ settype(j), setname(j),
         "$ "", ",", ",", 0,ec )
            if( l .gt. 0 ) then
               ival = igtvl( label(j), "", ec )
            write(".*", 'got GDB ",label(j),ival)
            tval = ival
            if( l .eq. 4 ) then
               ival = i4swap( ival )
            elseif( l .eq. 2 ) then
               ival = i2swap( ival )
            endif
         else
            rval = rgtvl( label(j), "", ec )
call vaximb( rval,ival,1 )
        endif
        iother(j) = ival  ! remember current SEGY pattern
    endif
   else  ! it is a channel
      write(".*", 'get CHAN ",object(j),settype(j)
      if( l .gt. 0 ) then
         ival = other(j)  ! get DFCRVR'ed value
      tval = ival
      if( l .eq. 4 ) then
         ival = i4swap( ival )
      elseif( l .eq. 2 ) then
      endif
   endif
   100 continue
   write(".*", 'read file ",object(j),settype(j)
   call filefdn( label(j), object(j),
   "$ settype(j), setname(j),
   "$ "", ",", ",", 0,ec )
   if( l .gt. 0 ) then
      rval = rgtvl( label(j), "", ec )
call vaximb( rval,ival,1 )
   endif
   endif
endif
endif
ival = i2swap( ival )
endif
else
   rval = other(j)        ! get DFCRVR'ed value
   call vaxibm( rval,ival,1 )
endif
endif
d    write(*,1010) ival, jval, j, index(j), l, object(j)
d1010    format( 'hdr_val=',z10,i10,' @',i3,' to ',i3,':',i1,x,a )
    call moveb( ival, thdr(index(j)), abs(l) )
endif
100   continue
c
   return
c
end

c

c
c
c
.title  vaxibm

;+  
; call vaxibm( from, to, n )  
;
; from = real array of vax fp data
; to = byte array of ibm floating data in tape order
; n = number of samples
;
; vax 780 timing, about 32 usec per sample
;-

.entry  vaxibm, ^m<r2,r3,r4,r5>

moval  @4(ap), r0  ; addr of in
moval  @8(ap), r1  ; addr of out
movl  @12(ap), r2  ; value of n
bleq  70$  ; if none to be done

ashl  #2,r2,r3  ; convert to bytes
addl  r3,r0  ; work from the top down
addl  r3,r1  ; makes byte swap easier
10$:
  rotl  #16,-(r0), r3  ; word swap float into register
movl  r3,r5
beql  40$  ; br if identically zero

extzv  #23, #8, r3, r4  ; get the exponent
subl  #128, r4  ; unbias it
biclt3  #^xff800000,r3,r5  ; get the mantissa; zap sign bit
bisl  #^x00800000,r5  ; make the hidden bit explicit
20$:
  bitl  #3,r4  ; make sure low bits won't get
  beql  30$  ; zapped by divide by 8

incl  r4
ashl  #-1,r5,r5
brb  20$
30$:
  ashl  #-2,r4,r4  ; binary to hex exponent
addl  #64, r4  ; put ibm bias in
insv  r4, #24, #8, r5  ; plug in exp to mantissa
biclt  #^x7fffffff, r3  ; kill all but the sign
bisl  r3, r5  ; glue sign on to mant and exp
40$:
movl  r5, -(sp)  ; make ibm byte order to output
.rept 4  ; this was the reason to work
movb  (sp)+,-(r1)  ; top down...
.endr
sobgtr  r2, 10$  ; until done
70$:
ret

.end

c
c
c
dmpgdb.for
c
c from LIS/A "How to Use LIS/A" appendix G
c very slightly changed, and without comments
c
subroutine dmpgdb( lunout )
  integer lunout

c
  include 'lisa:lisdef.inc'
  include 'lisa:except.inc'
  include 'lisa:lisedt.inc'

c
  integer icx(scxt),i,j,k,l,m,n,o,index,ec,rcode,count,lu
  integer igtlrp,igtct,igtlu
character*(slabel) settyp,setnam,objnam,atb,units
character*(slabel) cgttyp,cgtsnm,cgtonm,cgtlb,cgtut
character*(20) cval, cgsvl

c
c
call savcxt( icx )
write(lunout,*)
write(lunout,*) 'start',-----------------------------------gdb-----------------------------------'
c
do 600 i=1,kloopf
  call nxtlu( ec )
  if( iand(ec,mcc) .ne. ccsucc ) goto 700
  lu = igtlu(ec)
write(lunout,*) 'lu number=',lu
  do 500 j=1,kloopf
    call nxtrel( ec )
    if( iand(ec,mcc).ne.ccsucc) goto 600
    call dishrr(ec)
    do 400 k=1,kloopf
      call ntxtap( ec )
      if( iand(ec,mcc).ne.ccsucc) goto 500
      call disthr(ec)
      do 300 l=1,kloopf
        call nxtfil( ec )
        if( iand(ec,mcc) .gt. ccsucc ) goto 400
        call disfhr( ec )
        do 200 m=1,kloopf
          call nxtset( ec )
          if( iand(ec,mcc) .gt. ccsucc ) goto 300
  500 continue
  do 600 continue
setnam = cgtsnm( ec )
settyp = cgttyp( ec )
write(lunout,*) 'setnam=',setnam
write(lunout,*) 'settyp=',settyp

do 100 n=1,kloopf
   call nxtobj( ec )
   if( iand(ec,mcc) .gt. ccsucc ) goto 200
   objnam = cgtobjm( ec )
   write(lunout,*) 'object=',objnam
   do 50 a=1,kloopf
      call nxtabl( ec )
      if( iand(ec,mcc) .gt. ccsucc ) goto 100
      atb = cgtlb( ec )
      units = cgtut( '"',ec )
      rcode = igtlrp( '"',ec )
      count = igtct( '"',ec )
      do 25 index=1,count
         cval = cgtvi( '"',ec )
         write(lunout,*) atb,units,rcode,count,index,
            cval
            25         continue
      50         continue
      100        continue
      200        continue
      300        continue
      400        continue
      500        continue
      600        continue
      700        continue
      call rstct( icx )
      write(lunout,*) 'end ','----------gdb----------'
   c
      return
   end

   subroutine dishr( lunout, ec )
   integer lunout
   integer ec
   include 'lisa:lisdef.inc'
   include 'lisa:except.inc'

   integer icx(scxt)
   character*74 cgtriv
ec = ccsucc

    call savctx(icx)
    call getthr(ec)

if( iand(ec,mcc) .le. ccwarn) then

    write(lunout,*) '--------reel header--------'
    write(lunout,'(17x,a,a8)') 'service name=',cgtvl('SNAM','"',ec)
    write(lunout,'(17x,a,a8)') 'date=',cgtvl('DATE','"',ec)
    write(lunout,'(17x,a,a8)') 'origin=',cgtvl('ODAT','"',ec)
    write(lunout,'(17x,a,a8)') 'reel name=',cgtvl('RNAM','"',ec)
    write(lunout,'(17x,a,a8)')
    $    'reel continuation #=',cgtvl('RCON','"',ec)
    write(lunout,'(17x,a,a8)') 'reel previous #=',cgtvl('PNAM','"',ec)
    write(lunout,'(17x,a,a74)') '"',cgtvl('CMNT','"',ec),"

write(lunout,*) '--------reel header--------'
endif

    call rstctx(icx)
    return
end


subroutine disthr( lunout, ec)
integer lunout
integer ec

include 'lisa:lisdef.inc'
include 'lisa:except.inc'

integer icx(scxt)
character*74 : cgtvl

ec = ccsucc

    call savctx(icx)
    call getthr(ec)

if( iand(ec,mcc) .le. ccwarn) then

    write(lunout,*) '--------tape header--------'
    write(lunout,'(17x,a,a8)') 'service name=',cgtvl('SNAM','"',ec)
    write(lunout,'(17x,a,a8)') 'date=',cgtvl('DATE','"',ec)
    write(lunout,'(17x,a,a8)') 'origin=',cgtvl('ODAT','"',ec)
    write(lunout,'(17x,a,a8)') 'tape name=',cgtvl('TNAM','"',ec)
    write(lunout,'(17x,a,a8)')
    $    'tape continuation #=',cgtvl('TCON','"',ec)
write(lunout,'(17x,a,a8)') 'tape previous #=',cgtvl('PNAM','"",ec)
c  write(lunout,"') '-------tape header-------'
  endif
  call rstcxt( icx )
  return
end
c

subroutine disfhrl( lunout, ec )
  integer lunout
  integer ec
c
  include 'lisa:lisdef.inc'
  include 'lisa:except.inc'
c
  integer icx(scxt)
  character*10 cgtvl
c
c  ec = ccsucc
  call savcxt( icx )
  call getfhrl( ec )
  if( iand(ec,mcc) .le. ccwarn) then
c    write(lunout,"') '-------file header-------'
      write(lunout,'(17x,a,a10)') 'file name=',cgtvl('FNAM','"",ec)
      write(lunout,'(17x,a,a10)') 'service name=',cgtvl('SSUB','"",ec)
      write(lunout,'(17x,a,a10)') 'version=',cgtvl('VNUM','"",ec)
      write(lunout,'(17x,a,a10)') 'date=',cgtvl('DATE','"",ec)
      write(lunout,'(17x,a,a10)') 'max len=',cgtvl('MPRL','"",ec)
      write(lunout,'(17x,a,a10)') 'file type=',cgtvl('FTYP','"",ec)
      write(lunout,'(17x,a,a10)') 'previous file=',cgtvl('PNAM','"",ec)
c    write(lunout,"') '-------file header-------'
  endif
  call rstcxt( icx )
c
  return
end
$
USER AGREEMENT
DEC VAX COMPUTING ENVIRONMENT

BETWEEN: University of Texas Institute for Geophysics, with a principal place of business at Austin, Texas (hereinafter LICENSEE);

AND

SCHLUMBERGER WELL SERVICES, 5000 Gulf Freeway, P. O. Box 2175, Houston, Texas 77252-2175 (hereinafter SCHLUMBERGER).

WHEREAS SCHLUMBERGER and its Affiliates have developed a computer program, including a library of software routines, for accessing and handling wireline logging data tapes for the DEC VAX computing environment (hereinafter LIS/Access).

AND, WHEREAS LICENSEE wishes to receive and have the right to use this program, and SCHLUMBERGER agrees to make this program available to LICENSEE.

The parties have agreed to the following:

1. LICENSEE agrees that it is receiving from SCHLUMBERGER for its internal use a copy of LIS/Access. The copy, together with all modifications and supplements and all related information supplied now or in the future to LICENSEE, is hereinafter referred to as the Program. The Program shall also include any non-DEC ported versions of LIS/Access and all related information provided to LICENSEE now or in the future by third parties under an obligation of confidence to SCHLUMBERGER. Such software and related information may be provided in the form of executable software code, listings, and documentation. In the event that LICENSEE desires to obtain the Program source code, in order to port the Program to a non-VAX computing environment or for other reasons, subject to the approval of SCHLUMBERGER, a separate license agreement shall be required.

2. In return for such delivery, LICENSEE agrees to provide SCHLUMBERGER all modifications and supplements to the Programs which LICENSEE may develop in the future. Such modifications and supplements shall be considered SCHLUMBERGER confidential information and referred to as the Program under this agreement, subject to LICENSEE's right to use of such modifications and supplements. LICENSEE shall also fully report to SCHLUMBERGER its use of the Program and the results of all evaluations and tests of the Program. Supplements shall include any conversion software or product in SEG Y format.

3. SCHLUMBERGER MAKES NO WARRANTY OF ANY KIND WITH RESPECT TO THE PROGRAM, AND DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. SCHLUMBERGER disclaims all liabilities or obligations on the part of SCHLUMBERGER for damages, including but not limited to special, indirect, or consequential damages, arising out of or in connection with the use or performance of the Program, and LICENSEE agrees that SCHLUMBERGER shall not have any such liabilities or obligations.

4. LICENSEE acknowledges that SCHLUMBERGER has a proprietary interest in the Program, and agrees to hold such Program in confidence and acknowledges that such Program is made available to LICENSEE on a non-exclusive and non-transferable basis. LICENSEE agrees to use the Program only on its own CPU's and not to sell, disclose or otherwise make the Program available to others in any form or media. Further, neither
LICENSEE nor its representative shall divulge any information with respect to the technology embodied within the Program.

5. LICENSEE shall take appropriate action by instruction, agreement, bonding, or otherwise with its employees or any person permitted access to the Program to satisfy LICENSEE's obligations under Paragraph 4 of this agreement. Such action shall not be less than that taken by LICENSEE to protect its own property which it considers to be confidential and/or proprietary. In addition, no third party shall be permitted access to the Program without the express written consent of SCHLUMBERGER. LICENSEE agrees to reproduce and include SCHLUMBERGER's copyright and/or proprietary notice on all copies of the Program including all modifications.

6. The term of this agreement shall be indefinite provided LICENSEE is in compliance with all the terms and conditions of this agreement. If LICENSEE fails to comply with its obligations under Paragraphs 4 and 5 of this agreement, this agreement can be terminated by SCHLUMBERGER upon notice to LICENSEE. In the case of termination, the right of LICENSEE to use the Program shall cease and LICENSEE shall destroy or return to SCHLUMBERGER all copies of the Program or of part of the Program or of software derived from the Program, and information relating to the program, in whatever form.

7. LICENSEE agrees that this User Agreement shall not be subject to any terms or conditions contained in any standard or special purchase order form or similar document, whether such form is received before or after execution of this agreement or delivery of the Program.

8. LICENSEE's rights and obligations as set forth in this Agreement are hereby extended to LICENSEE's affiliates which agree to be bound by this Agreement. A LICENSEE affiliate is defined as any business entity: a) which is controlled by LICENSEE; b) which controls LICENSEE; or, c) which is controlled by the business entity controlling LICENSEE. For the purpose of this definition, "control" means the ownership, directly or indirectly, of more than 50% of the voting stock or of the ownership interest.

SCHLUMBERGER WELL SERVICES

Signed Calvin S. Wight
Name Calvin S. Wight
Title SR. STAFF ENGINEER
Date October 13, 1987

University of Texas Institute for Geophysics

Signed Joseph N. Phillips
Name Joseph D. Phillips
Title RESEARCH SCIENTIST
Date 14 October 1987
Mr. Calvin S. White  
Schlumberger Well Services  
5000 Gulf Freeway, P.O. Box 2175  
Houston, Texas 77001

Dear Mr. White:

Enclosed is a signed copy of the LIS/Access User Agreement which will enable scientists of the University of Texas Institute for Geophysics (UTIG) to develop software for SEG-Y conversion of the LIS format seismic waveform data acquired in Vertical Seismic Profile (VSP) experiments aboard the Ocean Drilling Program's (ODP) ship JOIDES RESOLUTION.

In regard to paragraphs 2 and 4 of the subject agreement referring to the LICENSEE's use of the LIS/SEG-Y conversion software, please note it is the intent of UTIG to make the conversion software available only to those scientists who wish to convert LIS seismic waveform data generated by VSP experiments aboard the ODP ship JOIDES RESOLUTION to SEG-Y. We understand that such a use of the conversion software meets with your approval. Would you please confirm this by returning a signed copy of this letter.

In the meantime, I wish to thank you for the help that you have provided in expediting our SEG-Y conversion project for the ODP. We now look forward to working with Don Wilhelmsen and Ted Spalding of your Austin Systems Center to accomplish the task.

Sincerely,

Joseph D. Phillips  
Research Scientist

cc: Dr. Arthur E. Maxwell  
    Dr. Thomas A. Davies

Acknowledged:

Date:
October 29, 1987

Dr. Joe D. Phillips
Institute for Geophysics
8701 Mopac Blvd.,
Austin, TX 78751-2789

Dear Dr. Phillips:

We have received your letter of October 14, 1987, along with the signed copy of the LIS/A User Agreement. Schlumberger understands your intended usage of the software aboard the Ocean Drilling Project ship JOIDES RESOLUTION and agrees that this usage is within the scope of the conditions of the User Agreement.

It must be made clear, however, that Schlumberger's agreement to this usage in no way relieves UTIG, as LICENSEE, of the obligations of the User Agreement and specifically the confidentiality provisions set forth in paragraphs 4 and 5 of the User Agreement.

If there are further questions please feel free to contact me.

Sincerely,

Calvin S. White

cc: Henry Garrana