Durham University

Durham E-Theses

Processing of a multichannel seismic reflection survey in the Hebridean region with special emphasis on improvements in velocity analysis

Richard William Hobbs

How to cite:

Hobbs, Richard William (1985) Processing of a multichannel seismic reflection survey in the Hebridean region with special emphasis on improvements in velocity analysis. Doctoral thesis, Durham University.

Use policy

The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a https://etheses.durham.ac.uk/id/eprint/7614/ is made to the metadata record in Durham E-Theses
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

Please consult the full Durham E-Theses policy for further details.

Academic Support Office, The Palatine Centre, Durham University, Stockton Road, Durham, DH1 3LE
e-mail: e-theses.admin@durham.ac.uk Tel: +44 0191 334 6107
https://etheses.durham.ac.uk

Processing of a Multichannel Seismic Reflection Survey in the Hebridean Region with Special Emphasis on Improvements in Velocity Analysis

by

Richard William Hobbs

Volume II

A thesis submitted for the degree of
Doctor of Philosophy at the University of Durham

Graduate Society

August, 1985

APPENDIX 2

CONTENTS

DURHAM SEISMIC PROCESSING SYSTEM SOFTWARE

SEG-A Demultiplex Program Description	A2-1
SEG-A Program Listing	A2-3
SEG-C Demultiplex Program Description	A2-31
SEG-C Program Listing	A2-33
Sort Program Description	A2-52
Sort Program Listing	A2-82
Hebridean Modification	A2-101
Time Sequence Analysis Program Description	A2-105
Time Sequence Analysis Program Listing	A2-168
Plot Program Description	A2-229
Plot Program Listing	A2-250
Velocity Analysis Program Description	A2-279
Velocity Analysis Program Listing	A2-310
Utility Subroutines Description and Listings	A2-353

To demultiplex SEG-A data recorded on the SDS-1010, i.e. Durham data recorded prior to 1984, use the following proceedure.

1) Load RK05 SEG-A scratch disc.

2) Create or edit parameter file DK6:DMXMIN.DAT

```
NFILES,  NCHAN,   ITSIZ, IHSTRT,  TPDRR,  TPDRW
IERFLG, NRECOV,   NUERR,  NALOW
(ICHAN(I), I=1,NCHAN)
LINENO,  NSAMP,  IGUNNO, SLSPAC, IWIDTH,  SROFF, ITOTAL
AREANO
DAYNO
YEARNO
OBSVNO
```

IFILES (I6) Number of files to demultiplex

NCHAN (I6) Number of channels to demultiplex from each file

ITSIZ (I6) Number of 128 data sample blocks to demultiplex for each channel

IHSTRT (I6) Number of 128 sample block from which to start output

TPDRR (I6) Tape read drive number

TPDRW (I6) Tape write drive number

IERFLG (I6) Demultiplex flag

0 = fast demultiplex with minimum error checking
1 = slow demultiplex with full error checking

NRECOV (I5) Data recovery on error when in fast demultiplex mode

0 = no recovery, blank output file
1 = try slow demultiplex with full error checking

NUERR (I5) Number of errors allowed per trace

NALOW (I5) Total number of errors allowed in file

ICHAN(I), I=1,NCHAN
(12I5) List of receiver array channel numbers to be demultiplexed and output to tape.

LINENO (I5) Line number

NSAMP (I5) Number of samples in output traces

IGUNNO (I5) Number of guns used in source array

SLSPAC (I5) Shot spacing in metres

IWIDTH (I5) Width of source array in metres

SROFF (I5) Source receiver offset in metres

ITOTAL (I5) Total length of receiver array in metres

AREANO (24A1) Area name

DAYNO (5A1) Jul-ian day number

YEARNO (6A1) Year

OBSVNO (18A1) Observer name

3) Load input and output tapes, set tape density switches to remote.

4) Run DK6:DWREAD to check input tape format.

5) Rewind input tape.

6) Run DK1:DEMUXA or DK6:DEMUXA to start program

7) The program will prompt for the following extra information

GIVE CURRENT OUTPUT REEL NUMBER give reel sequence number in line.

GIVE NEXT OUTPUT FILE NUMBER give the number of the next shot gather, this may not be the same as the field record number recorded by the SDS-1010 due to missing files.

GIVE NEXT TRACE SEQUENCE NUMBER give trace sequence number in line.

ENTER NO OF ZERO FILES TO BE ADDED give the number of files lost during tape change.

8) The program will run until either the read or write tapes end, when a prompt will be issued to get new drive number. The program will automatically set the read drive to 800 NRZI format and the write drive to 1600 PE format.

9) A simple log file is written to DK6:DMXMIN.LOG

```
C        DEMULTIPLEX PROGRAM
C
C   THIS PROGRAM TAKES DATA FROM TAPE(VIA PDP-11),
C  IN THE MULTIPLEXED FORMAT USED AT DURHAM UNIV (SEG A)
C  AND DEMULTIPLEXES THE CHANNELS AND REFORMATS THE DATA
C  VALUES. THESE ARE THEN WRITTEN TO DISC AND COPIED TO
C  TAPE(VIA PDP-11) FOR STORAGE IN SEG Y FORMAT (WITH TAPE MARKS)
C
C
C       DATA STORAGE DECLARATIONS.
C
C       ____________________________
      VIRTUAL BUFF(8448)
      REAL*8 FNAMW,DBLK(2)
      REAL*4 DISCHN,BUFOUT(256),TAPEHN,BUFCLR(128)
      INTEGER*4 JFNUM,JORFNO,JORTNO
      INTEGER*2 CHOFF(30),NBLKOF(30),IBLKOF(30),DISKAD,BSST,
     $EGAINS(30),GCNT,DMXBUF(3840),SPOS,FST,BSST1,BLST,OLAP,
     $RST,ICHAN(30),GSAVE(30),MASK,SYNC,BUFF,HSBLKW(5),
     $GAINS(3840),STATUS,TPDRR,TPDRW,LINENO,
     $REELNU,NCHAN,NSAMP,IGGUNO,SLSPAC,IWIDTH,SROFF,ITOTAL
      LOGICAL*1 HSBLK(240),OUTBUF(16624),AREANO(24),DAYNO(5),
     $YEARNO(6),OBSVNO(18),HBLKA(10),RC
      COMMON /DATBUF/OUTBUF
      COMMON /REELHD/LINENO,REELNU,NSAMP,IGUNNO,SLSPAC,
     $IWIDTH,SROFF,ITOTAL,AREANO,YEARNO,OBSVNO,DAYNO
      COMMON /SUBS/GSAVE,IFDIR,IERR,IERFLG,SPOS,RC
      COMMON /DECOM/CHOFF,ICHAN,NCHAN,NBLKOF,GCNT,IHSTRT,IHSEC
      COMMON /BUFCOM/BSST1,BLST,OLAP,RST,DISKAD,ITSIZ
      COMMON/BUFS/NUERR,NALOW,ITIC,IERSYN,IERCOM,IERTCK,
     $IERGCK,IERGDR
      EQUIVALENCE (HSBLKW(1),HBLKA(1)),(HSBLK(9),JFNUM),
     $(HSBLK(181),JORFNO),(HSBLK(185),JORTNO),(HSBLK(115),ISAMP),
     $(HSBLK(117),ITIME),(HSBLK(109),IDELAY),(HSBLK(29),ITYPE)
      EQUIVALENCE (HSBLK(1),OUTBUF(1)),(GAINS(1),OUTBUF(241)),
     $(DMXBUF(1),OUTBUF(7921)),(BUFOUT(1),OUTBUF(15601))
      DATA APHAN/3RAP /,TAPEHN/3RTM /
      DATA CHOFF/4352,4480,4608,4736,4864,4992,
     %5120,5248,5376,5504,5632,5760,5888,6016,6144,6272,
     %6400,6528,6656,6784,6912,7040,7168,7296,7424,7552,
     %7680,7808,7936,8064/
      DATA FNAMW/12RDK6DEMPLXDAT/
      DATA MASK,SYNC/"17,"177777/
      DATA BUFCLR/128*0.0/
C
C CONSTANTS USED IN PROGRAM
C
      DO 5 I=1,240
    5 OUTBUF(I)=0
      IPADNO=0
      IEOTR=0
      ILEN=0
      IEOTW=1
      BSST=4096+256
      IBIAS=15
      IAD=BSST+1
C
C  SET UP THE RT-11 INPUT-OUTPUT PROCEDURES
C
      IF(IFETCH(TAPEHN).NE.0) STOP 'TAPEHN BAD FETCH'
      IF(IFETCH(APHAN).NE.0) STOP 'APHAN BAD FETCH'
C
C  INITIALIZE THE AP
C
      CALL APINIT(0,0,I)
C
C  READ IN THE NECESSARY INPUT DATA
C
      OPEN(UNIT=1,NAME='DK6:DMXMIN.DAT',TYPE='OLD',ERR=300)
      GOTO 310
  300 STOP'NO DATA FILE'
  310 READ(1,1001)NFILES,NCHAN,ITSIZ,IHSTRT,TPDRR,TPDRW
```

```
 1001 FORMAT(12I5)
      READ(1,1001)IERFLG,NRECOV,NUERR,NALOW
      READ(1,1001)(ICHAN(I),I=1,NCHAN)
C
C READ IN HEADER INFO
C
      READ(1,1001) LINENO,NSAMP,IGUNNO,SLSPAC,IWIDTH,SROFF,ITOTAL
      READ(1,1003) AREANO,DAYNO,YEARNO,OBSVNO
 1003 FORMAT(24A1/5A1/6A1/18A1)
      CLOSE(UNIT=1)
      TYPE 2002
 2002 FORMAT('$GIVE CURRENT OUTPUT REEL NO.')
      ACCEPT 1001,REELNU
      TYPE 2010
 2010 FORMAT('$GIVE NEXT OUTPUT FILE NO.')
      ACCEPT 1001,IFNUM
      TYPE 2012
 2012 FORMAT('$GIVE NEXT TRACE SEQUENCE NO.')
      ACCEPT 2014,FTRCNO
 2014 FORMAT(F15.0)
C
C   SET UP CONSTANTS AND REST OF RT-11
C   INPUT OUTPUT ROUTINES.
C
      ITSIZO=ITSIZ-IHSTRT+1
      NWRD=ITSIZO*256
      IBLKOF(1)=1
      DO 10 J=2,NCHAN
   10 IBLKOF(J)=IBLKOF(J-1)+ITSIZO
      IFSIZ=(ITSIZO*NCHAN)+1
      IF(IENTER(4,FNAMW,IFSIZ).LT.0)STOP' ENTER ERROR'
C
C  WRITE LOG FILE HEADER BLOCK
C
      OPEN(UNIT=1,NAME='DK6:DMXMIN.LOG',TYPE='NEW',ERR=330)
      GOTO 340
 330  STOP'ERROR OPENING LOG FILE'
 340  WRITE (1,1999)
 1999 FORMAT(' ERROR STATISTICS',/,' ***************',/,
     $'      FILE      SYNC BLOCK  COMMUNICATION  TIME CODE',
     $'    GAIN CODE       GAIN      H/BLK FILE',/,'    NUMBER',
     $'       LOSSES       LOSSES        LOSSES       LOSSES',
     $'      DIRECTION     NUMBER')
C
C LOOKUP WRITE TAPE DRIVE AND WRITE HEADER BLOCKS IF AT BOT
C
      CALL REELHD(TPDRW,IFNUM,IEOTW)
C
C    START OF THE MAIN DEMULTIPLEX LOOP
C
      DO 999 I=1,NFILES
      ITRY=0
      IERR=0
C
C SEE IF FILES TO BE ZEROED
C
      IF(IPADNO.GT.0) WRITE(1,1020) IFNUM
      IF(IPADNO.GT.0) GO TO 230
C
C CHECK ARNT AT EOT
C
      IF(I.EQ.1) GOTO 41
      IF(IEOTR.EQ.0)GOTO 43
      WRITE(7,1061)
 1061 FORMAT(' ENTER NEW READ DRIVE NO:',$)
      READ(5,1062)TPDRR
 1062 FORMAT(I1)
 41   WRITE(1,2001) TPDRR,IFNUM
 2001 FORMAT(' START READ DRIVE :',I1,' FILE NUMBER :',I4)
      IEOTR=0
      IF(TPDRR.GT.2)GOTO 265
C
```

```
C CHECK IF ZERO FILES TO BE ADDED AT END OF TAPE
C
      WRITE(7,1063)
 1063 FORMAT(' ENTER NO OF ZERO FILES TO BE ADDED(I2):',$)
      READ(5,1064)IPADNO
 1064 FORMAT(I2)
C
C  FILE ORGANISATION ON A NORMAL RUN
C
  43  ITRY=ITRY+1
      IERR=0
      IERSYN=0
      IERCOM=0
      IERTCK=0
      IERGCK=0
      IERGDR=0
      IF(IPADNO.GT.0)GOTO 100
      ITYPE=1
C
C  READ FIELD TAPE FROM SYSTIME DECK
C
      DO 35 M=1,NCHAN
   35 NBLKOF(M)=IBLKOF(M)
      CALL EXREAD(1,TPDRR,STATUS,0,IEOTR,ILEN)
      IF(IEOTR.EQ.0) GOTO 37
      IF(IEOTR.LT.0) GOTO 38
      TYPE 1050,TPDRR,IFNUM
 1050 FORMAT(' EOT ENCOUNTERED ON READ DRIVE:',I1,' FILE NO:',I4)
      GOTO 37
   38 IFNUM=IFNUM-1
      TYPE 1049,TPDRR,IFNUM
 1049 FORMAT(' LEOT ENCOUNTERED ON READ DRIVE:',I1,' FILE NO.:',I4)
      GOTO 999
   37 IF(STATUS.EQ.0
     $          .AND.ILEN.GE.ITSIZ) GO TO 45
C
C  CHECK STATUS WORD
C
      IF(STATUS.EQ.-9) STOP 'WRONG DISC FORMAT'
      WRITE(1,1051) STATUS,TPDRR,IFNUM,ILEN
      TYPE 1051,STATUS,TPDRR,IFNUM,ILEN
 1051 FORMAT(' STATUS:',I3,' DRIVE NO:',I1,' FILE NO:',I4,
     $' 1/2 SEC BLK LENGTH',I5)
      IF(STATUS.EQ.-8) PAUSE'READ DRIVE NOT AVAILABLE'
      IF(STATUS.EQ.-8) GOTO 43
      IF(ITRY.GE.3
     $        .AND.ILEN.GE.ITSIZ) GO TO 100
      IF(ITRY.GE.3)STOP'SHORT READ FILE'
C
C  REWIND FILE AND TRY TO READ AGAIN
C
      CALL EXREAD(0,TPDRR,STATUS,-1,IEOTR,ILEN)
      IF(STATUS.NE.0) TYPE 1001,STATUS
      IF(STATUS.NE.0) STOP 'READ TAPE ERROR'
      GO TO 43
C
C START OF MAIN BUSINESS
C
  45  NERR=0
      DISKAD="122000
      GCNT=0
      IHSEC=0
      RC=.FALSE.
C
C   READ IN FIRST DATA BLOCK FROM DISC AND EXTRACT THE
C   HEADER INFORMATION
C
      CALL DISKRD(BUFF(1),8448,DISKAD,NERR)
      IF(NERR.LT.0) STOP 'READ ERROR'
C
C  END OF INITIAL READS
C  BEGINNING OF HEADER BLOCK SCANS
```

```
C
      DO 50 LL=1,4
 50   HSBLKW(LL)=BUFF(LL).OR."30060
      HSBLKW(5)=BUFF(5)
      ITIC=2*HBLKA(9)
      IGJL=1
      DO 60 JL=1,30
      EGAINS(JL)=(IBTSWP(BUFF(5+JL))).AND.MASK
      GSAVE(JL)=EGAINS(JL)+IBIAS
      GAINS(IGJL)=GSAVE(JL)
      IGJL=IGJL+128
 60   CONTINUE
      SPOS=35
 70   SPOS=SPOS+1
      IF(BUFF(SPOS).EQ.0)GO TO 70
C
C   BUFFER CONSTANTS SET UP AFTER END OF HEADER BLOCK IS LOCATED
C
      FST=SPOS
      BSST1=4096+FST
      BLST=8192+FST
      OLAP=257-FST
      RST=4096-OLAP
C
C  CHECK SYNC WORDS FOR ERRORS AND EXTRACT THE GAINS READY
C  FOR USE AS INTEGERS.
C
      IFDIR=IDIRG(IBTSWP(BUFF(SPOS+1)))
      IF(IERFLG.NE.0.OR.IERR.NE.0)CALL BUFSCN(BUFF,0,IFNUM)
      IF(IERR.LT.0)GOTO 100
      IF(IERFLG.NE.0.OR.IERR.NE.0)GOTO 220
C
C  PUT FIRST BUFFER INTO AP
C
      CALL APPUT(GSAVE,4096,30,1)
      CALL APPTXM(BUFF(FST),0,4096,1)
C-------------------------------------------------------------
C   MAIN DEMUX LOOP FOR A SINGLE FILE
C
C-------------------------------------------------------------
C
C  COME HERE IF WANT TO DEMUX ALL THE FILE
C WITHOUT FULL ERROR CHECKING SWITCHED ON

      CALL DMX(BUFF)
      IF(IERR.EQ.0) GO TO 230
 215  WRITE(1,1030)IFNUM
1030  FORMAT(' ERROR ON FILE:',I4,' FOUND GOING INTO RECOVERY MODE')
      IF(NRECOV.EQ.0)GOTO 100
      DO 216 LLZ=1,NCHAN
 216  NBLKOF(LLZ)=IBLKOF(LLZ)
      GOTO 45
C
C  COME HERE IF WANT TO DEMUX ALL THE FILE
C WITH FULL ERROR CHECKING SWITCHED ON
C
 220  CONTINUE
      IHSEC=IHSEC+1
      IF (IHSEC.GE.IHSTRT)CALL DEMUX
      IF(IHSEC.GE.ITSIZ)GOTO 230
      CALL BUFSCN(BUFF,1,IFNUM)
      IF(IERR.GE.0)GOTO 220
C-------------------------------------------------------------
C   END OF MAIN DEMUX LOOP
C-------------------------------------------------------------
C
C BLANK PARTS OF FILES WITH FATAL ERRORS
C
 100  WRITE(1,1020)IFNUM
1020  FORMAT(' FILE NUMBER ',I4,' BLANKED')
      ITYPE=3
      DO 125 JZ=1,NCHAN
```

```
      IBLK=IBLKOF(JZ)
      DO 125 LZ=1,ITSIZO
      IF(IWAIT(4).LT.0) STOP 'CLEAR ERROR'
      CALL IWRITE(256,BUFCLR,IBLK,4)
      IBLK=IBLK+1
  125 CONTINUE
C
C SET UP TRACE HEADER BLOCK
C
  230  CALL JICVT(IFNUM,JFNUM)
       CALL JJCVT(JFNUM)
       DECODE(5,2020,HBLKA(4),ERR=55) IORFNO
 2020 FORMAT(I5)
       GO TO 56
  55   IORFNO=0
  56   CALL JICVT(IORFNO,JORFNO)
       CALL JJCVT(JORFNO)
       DECODE(3,2030,HBLKA(1),ERR=57) IORTNO
 2030 FORMAT(I3)
       GO TO 58
  57   IORTNO=0
  58   CALL JICVT(IORTNO,JORTNO)
       CALL JJCVT(JORTNO)
       ISAMP=NSAMP
       ITIME=4000
       IDELAY=(IHSTRT-1)*500
C
C WRITE OUT HEADER BLOCK TO DISC FILE
C
       CALL IWAIT(4)
       IF(IWRITW(4,HBLKA,0,4).LT.0)STOP'HSBLK ERROR'
C
C  WRITE OUT GATHER FILE TO TAPE
C
  260  CONTINUE
       CALL DMXOUT(IBLKOF,TPDRW,4,NWRD,FTRCNO,IFNUM,IEOTW)
       IF(IEOTW.EQ.0) GO TO 261
       REELNU=REELNU+1
       CALL REELHD(TPDRW,IFNUM,IEOTW)
       GO TO 260
  261  IPADNO=IPADNO-1
       IF(IPADNO.GT.0) GOTO 990
       IPADNO=0
       CALL CLOSEC(4)
C
C   WRITE OUT LOG FILE DATA AND OPEN NEW DISK FILE
C
       IF(IENTER(4,FNAMW,IFSIZ).LT.0) STOP 'ENTER END ERR'
  990  WRITE(1,2000) IFNUM,IERSYN,IERCOM,IERTCK,IERGCK,IERGDR,
      $(HBLKA(K),K=1,3)
 2000 FORMAT( 4X,I4,5(9X,I4),10X,3A1)
C
C  GO TO NEXT INPUT FILE IF REQU'D
C
  999  IFNUM=IFNUM+1
  265  CLOSE(UNIT=1)
       CALL APRLSE
       CALL RESET
       STOP'NORMAL TERMINATION'
       END
```

```
      SUBROUTINE DMX(BUFF)
      VIRTUAL BUFF(8448)
      INTEGER*2 BUFF,BSST,BLST,ISPOS,BSST1,RST,OLAP,DISKAD,
     $CHOFF(30),NBLKOF(30),GCNT,GAINS(3840),ICHAN(30),GSAVE(30)
      REAL*4 BUFOUT(256)
      LOGICAL*1 OUTBUF(16624),RC
      COMMON/DATBUF/ OUTBUF
      COMMON /BUFCOM/BSST1,BLST,OLAP,RST,DISKAD,ITSIZ
      COMMON /SUBS/GSAVE,IFDIR,IERR,IERFLG,ISPOS,RC
      COMMON /DECOM/CHOFF,ICHAN,NCHAN,NBLKOF,GCNT,
     #IHSTRT,IHSEC
      EQUIVALENCE (GAINS(1),OUTBUF(241)),(BUFOUT(1),OUTBUF(15601))
C
C    DO DEMUX AND BINARY SCALING
C
 5    IHSEC=IHSEC+1
      NIN=128
      NOUT=2*NIN
      CALL APWD
      CALL DMXA(4096,4352,128,GCNT,IFDIR,NIN)
      CALL APWR
      CALL APGSP(IERR,15)
      CALL APWD
      IF(IERR.NE.0)IERR=1
      IF(IERR.NE.0)RETURN
      CALL APGSP(GCNT,3)
      CALL APGSP(IFDIR,4)
      IF(IHSEC.LT.IHSTRT)GOTO 30
      NCHADD=CHOFF(ICHAN(1))
      CALL APGET(BUFOUT(1),NCHADD,NIN,2)
      IIN=129
      IOUT=1
      IF(NCHAN.EQ.1)GOTO 20
C
C  EXTRACT EACH WANTED CHANNEL AND PUT ON DISC
C
      DO 10 NJ=2,NCHAN
      NJ1=NJ-1
      NCHADD=CHOFF(ICHAN(NJ))
      CALL APWD
      IF(IWAIT(4).LT.0) STOP 'DMX ERROR'
      CALL APGET(BUFOUT(IIN),NCHADD,NIN,2)
      CALL IWRITE(NOUT,BUFOUT(IOUT),NBLKOF(NJ1),4)
      NBLKOF(NJ1)=NBLKOF(NJ1)+1
      IT=IIN
      IIN=IOUT
      IOUT=IT
   10 CONTINUE
   20 IF(IWAIT(4).LT.0) STOP 'DMX ERR'
      CALL APWD
      CALL IWRITE(NOUT,BUFOUT(IOUT),NBLKOF(NCHAN),4)
      NBLKOF(NCHAN)=NBLKOF(NCHAN)+1
   30 RC=.NOT.RC
      CALL APWD
      IF(IWAIT(4).LT.0) STOP 'DMX END ERR'
C
C NORMAL RETURN
C
      IF(IHSEC.GE.ITSIZ) RETURN
C
C  THIS ROUTINE CONTROLS THE DOUBLE BUFFERING SCHEME USED TO TAKE DATA
C  FROM DISC AND PUT IT IN TO THE AP WHEN IT IS NEEDED.
C
      IF(RC)CALL APPTXM(BUFF(BSST1),0,4096,1)
      IF(.NOT.RC)CALL APPTXM(BUFF(BLST),0,OLAP,1)
      IF(.NOT.RC)CALL APPTXM(BUFF(257),OLAP,RST,1)
      IFILL=257
      IF(.NOT.RC)IFILL=4353
      CALL DISKRD(BUFF(IFILL),4096,DISKAD,NERR)
      IF(NERR.LT.0) STOP 'DMX READ ERR'
      GO TO 5
      END
```

```
      SUBROUTINE BUFSCN(BUFF,ICODE,FDONE)
      VIRTUAL BUFF(8448)
      INTEGER*2 BUFF,GAINS(3840),GSAVE(30),SPOS,DMXBUF(3840),
     %FRAME(33),FDONE
      LOGICAL*1 IF(2),IBYT(2),IBYTE(2),FRAMEB(66),OUTBUF(16624),
     %IBSYNC,RC
      COMMON/DATBUF/ OUTBUF
      COMMON /SUBS/GSAVE,IFDIR,IERR,IERFLG,SPOS,RC
      COMMON/BUFS/NUERR,NALOW,ITIC,IERSYN,IERCOM,IERTCK,
     $IERGCK,IERGDR
      EQUIVALENCE (IWORD,IF(1)),(IWORDF,IBYT(1)),(IWORDB,IBYTE(1)),
     %(FRAME(1),FRAMEB(1))
      EQUIVALENCE (GAINS(1),OUTBUF(241)),(DMXBUF(1),OUTBUF(7921))
      DATA ISYNC/"177777/,IBIAS/15/
      DATA IBSYNC/"377/
      IF(ICODE.GT.0)GOTO 1
      ICHCK=1
      ICD=0
      LPINT=0
      FRAME(33)=IBTSWP(BUFF(SPOS+1))
      SPOS=SPOS+2
      ITBIAS=IBTSWP(BUFF(SPOS+24))/2*2
      ITCONT=0
      NERR=0
      GSAVE(30)=30
      CALL FRAMFL(BUFF,FRAME,1,33,-1)
C
C DATA CHECK AND GAIN PREPARATION SUBROUTINE
C
    1 IGSCHK=GSAVE(ICHCK)
      LDONE=0
    5 LDONE=LDONE+1
      IF(LPINT.GT.0)GOTO 100
      FRAME(1)=FRAME(33)
      CALL FRAMFL(BUFF,FRAME,2,33,ICD)
C
C SYNC TEST
C
    6 IF(FRAME(32).NE.ISYNC)GOTO 20
      IF(FRAMEB(66).EQ.IBSYNC)GOTO 30
      IERSYN=IERSYN+1
C
C TEST FOR TYPE OF DATA CORRUPTION
C
   20 IF(LDONE.EQ.128)GOTO 90
      IF(NERR.GT.NUERR)GOTO 160
      NERR=NERR+1
      NFER=0
   25 DO 40 I=1,32
      IWORDB=FRAME(I)
      IF(IWORDB.NE.ISYNC)GOTO 50
      IWORDB=FRAME(I+1)
      IF(IBYTE(2).NE.IBSYNC)GOTO 50
C
C DETECT PATTERN OF BYTES LOST
C
      IND=1
      DO 60 L=I+1,33
      FRAME(IND)=FRAME(L)
      IND=IND+1
   60 CONTINUE
      IF(ICD.EQ.3)GOTO 86
   65 CALL FRAMFL(BUFF,FRAME,IND,33,ICD)
      ICD=0
      GOTO 6
   50 IF(IBYTE(1).NE.IBSYNC)GOTO 70
      IWORDF=FRAME(I+1)
      IF(IWORDF.NE.ISYNC)GOTO 70
C
C ODD BYTE LOSS OR GAIN DETECTED
C
      IND=1
```

```
      IF(I.EQ.32)GOTO 85
      DO 80 L=I+2,33
      IWORDB=FRAME(L)
      IF(2)=IBYT(1)
      IF(1)=IBYTE(2)
      FRAME(IND)=IWORD
      IWORDF=IWORDB
      IND=IND+1
   80 CONTINUE
   85 IF(ICD.EQ.3)ICD=1
      IF(ICD.EQ.1)GOTO 65
   86 CALL FRAMFL(BUFF,FRAME,IND,33,2)
      ICD=3
      GOTO 6
   70 IF(IBYTE(2).NE.IBSYNC)GOTO 40
      IWORDF=FRAME(I+1)
      IF(IBYT(1).NE.IBSYNC)GOTO 40
C
C COMMUNICATION ERR POSSIBLE
C
      IERCOM=IERCOM+1
C
C CHECK TO SEE IF OK TO SEARCH FURTHER
C
C AHEAD IN ATTEMPTING TO REESTABLISH CONTACT
C
   40 CONTINUE
      IF(NFER.GT.NALOW)GOTO 160
      NFER=NFER+1
      FRAME(1)=FRAME(33)
      CALL FRAMFL(BUFF,FRAME,2,33,0)
      GOTO 25
C
C TIME CHECK SECTION
C
   30 ITN=FRAME(26)/2*2
      ITP=ITCONT.OR.ITBIAS
      IF(ITN.EQ.ITP)GOTO 90
      IF(NERR.GT.NUERR)GOTO 160
      IERTCK=IERTCK+1
      NERR=NERR+1
      INTVAL=(ITN-ITP)/ITIC
      IF(INTVAL.LT.0)GOTO 105
      LPINT=INTVAL
  100 IGPOS=LDONE
      DO 110 IVI=1,29
      DMXBUF(IGPOS)=1
      GAINS(IGPOS)=GSAVE(IVI)
      IGPOS=IGPOS+128
  110 CONTINUE
      GAINS(IGPOS)=30
      LDONE=LDONE+1
      LPINT=LPINT-1
      ITCONT=ITCONT+ITIC
      ICHCK=ICHCK+1
      IF(ICHCK.GT.30)ICHCK=1
      IF(LDONE.GE.128)GOTO 150
      IF(LPINT.GT.0)GOTO 100
  105 CONTINUE
   90 ITCONT=ITCONT+ITIC
C
C GAINS CORRECTION
C
      IGSAVE=IGCHK(FRAME(1),IDIR)+IBIAS
      IF(IGSAVE.EQ.IGSCHK)GOTO 120
      IERGCK=IERGCK+1
      IF(NERR.GT.NUERR)GOTO 160
      NERR=NERR+1
      GSAVE(ICHCK)=IGSAVE
  120 IF(IFDIR.EQ.IDIR)GOTO 130
      IERGDR=IERDGR+1
      IF(NERR.GT.NUERR)GOTO 160
```

```
      NERR=NERR+1
      IFDIR=IDIR
C
C NORMAL WORK
C
  130 ICHCK=ICHCK+1
      IF(ICHCK.GT.30)ICHCK=1
      IGPOS=LDONE
      DO 140 L=1,29
      DMXBUF(IGPOS)=IGAIN(FRAME(L+1),GSAVE(L),IFDIR)
      GAINS(IGPOS)=GSAVE(L)
      IGPOS=IGPOS+128
  140 CONTINUE
      GAINS(IGPOS)=30
      IGSCHK=GSAVE(ICHCK)
      IFDIR=-IFDIR
      IF(LDONE.LT.128)GOTO 5
  150 CALL APPUT(DMXBUF,4352,3840,1)
      RETURN
  160 IERR=-1
      RETURN
      END




      SUBROUTINE DEMUX
      INTEGER*2 CHOFF(30),GANADD,NBLKOF(30),
     %GAINS(3840),ICHAN(30),GSAVE(30),GCNT
      REAL*4 BUFOUT(256)
      LOGICAL*1 OUTBUF(16624),RC
      COMMON/DATBUF/ OUTBUF
      COMMON /SUBS/GSAVE,IFDIR,IERR,IERFLG,ISPOS,RC
      COMMON /DECOM/CHOFF,ICHAN,NCHAN,NBLKOF,GCNT,
     #IHSTRT,IHSEC
      EQUIVALENCE (GAINS(1),OUTBUF(241)),(BUFOUT(1),OUTBUF(15601))
C
C    DO DEMUX AND BRING IN THE SAVED GAINS
C
      NIN=128
      NOUT=2*NIN
      CALL APWD
      CALL VFLT(4352,1,4352,1,3840)
      CALL APWR
      CALL APPUT(GAINS,0,3840,1)
C
C  FORM THE DEMUXED NOS INTO R*4 RERPRESENTATION
C  FOR EACH CHANNEL IN TURN AND THEN WRITE THEM
C  OUT TO DISC. THIS IS DONE FOR 128 SAMPLES OF
C  EACH CHANNEL WHICH ARE EXPECTED TO BE IN THE A.P.
C
      CALL APWD
      DO 10 NJ=1,NCHAN
      NCHADD=CHOFF(ICHAN(NJ))
      GANADD=NCHADD-4352
      CALL VBINSC(NCHADD,1,NCHADD,1,GANADD,1,NIN)
      CALL APWR
      IF(IWAIT(4).LT.0) STOP 'DEMUX ERROR'
      CALL APGET(BUFOUT,NCHADD,NIN,2)
      CALL APWD
      CALL IWRITE(NOUT,BUFOUT,NBLKOF(NJ),4)
      NBLKOF(NJ)=NBLKOF(NJ)+1
   10 CONTINUE
      IF(IWAIT(4).LT.0) STOP 'DEMUX ERR'
      RETURN
      END
```

```
      SUBROUTINE REELHD(IDRV,IFNUM,IEOT)
C
C IDRV WRITE DRIVE NUMBER
C
C IFNUM CURRENT FILE NUMBER
C
C IEOT END OF TAPE FLAG
C
      REAL*8 MTNAM(3)
      LOGICAL*1 RLHDBL(80,40),CLIENT(24),LINE(12),AREA(23),
     %REELNO(11),DAY(4),YEAR(5),OBSVER(17),TRINRC(8),CDPFLD(10),
     %SMPINT(9),SPINTR(7),BITS(6),BYTES(7),RECFMT(8),RLFMT(8),
     %UNITS(10),SFLTPT(5),GFLTPT(5),SOURCE(12),SORCNO(8),SCPTIN(18),
     %SWIDTH(18),SRCPTN(27),SPDOFS(8),SPDDIS(8),GRPINT(20),GEOGPN(5),
     %PROC1(64),PROC2(64),PROC3(76),PROC4(76),PROC5(76),AREANO(24),
     %DAYNO(5),YEARNO(6),OBSVNO(18),RLHDCD(400)
      INTEGER*2 LINENO,REELNU,NCHAN,NSAMP,IGUNNO,SLSPAC,IWIDTH,
     %SROFF,ITOTAL,TRCNUM,AUXTRS,SAMPIN,ORSINT,NOSMPL,ORSAMP,
     %FMTCD,CDPNO,SORTCD,VERTSM,BINGAN,AMPREC,UNITNO,POLARY,
     %CHOFF(30),NBLKOF(30),ICHAN(30),GCNT,ITMERR(2)
      INTEGER*4 LNNOL,RLNOL
      COMMON/REELHD/LINENO,REELNU,NSAMP,IGUNNO,SLSPAC,
     %IWIDTH,SROFF,ITOTAL,AREANO,YEARNO,OBSVNO,DAYNO
      COMMON/DECOM/CHOFF,ICHAN,NCHAN,NBLKOF,GCNT,
     %IHSTRT,IHSEC
      EQUIVALENCE (RLHDBL(1,1),CLIENT(1)),(RLHDBL(9,2),LINE(1)),
     %(RLHDBL(26,2),AREA(1)),(RLHDBL(12,3),REELNO(1)),
     %(RLHDBL(41,3),DAY(1)),(RLHDBL(50,3),YEAR(1)),
     %(RLHDBL(64,3),OBSVER(1)),(RLHDBL(23,5),TRINRC(1)),
     %(RLHDBL(71,5),CDPFLD(1)),(RLHDBL(20,6),SMPINT(1)),
     %(RLHDBL(42,6),SPINTR(1)),(RLHDBL(56,6),BITS(1)),
     %(RLHDBL(74,6),BYTES(1)),(RLHDBL(21,7),RECFMT(1)),
     %(RLHDBL(45,7),RLFMT(1)),(RLHDBL(71,7),UNITS(1)),
     %(RLHDBL(29,8),SFLTPT(1)),(RLHDBL(52,9),GFLTPT(1)),
     %(RLHDBL(17,11),SOURCE(1)),(RLHDBL(41,11),SORCNO(1)),
     %(RLHDBL(63,11),SCPTIN(1)),(RLHDBL(63,12),SWIDTH(1)),
     $(RLHDBL(17,12),SRCPTN(1)),(RLHDBL(19,15),SPDOFS(1)),
     $(RLHDBL(39,15),SPDDIS(1)),(RLHDBL(61,15),GRPINT(1)),
     $(RLHDBL(25,16),GEOGPN(1)),(RLHDBL(17,21),PROC1(1)),
     $(RLHDBL(17,22),PROC2(1)),(RLHDBL(5,23),PROC3(1)),
     $(RLHDBL(5,24),PROC4(1)),(RLHDBL(5,25),PROC5(1))
      EQUIVALENCE (RLHDCD(5),LNNOL),(RLHDCD(9),RLNOL),
     %(RLHDCD(13),TRCNUM),(RLHDCD(15),AUXTRS),(RLHDCD(17),SAMPIN),
     %(RLHDCD(19),ORSINT),(RLHDCD(21),NOSMPL),(RLHDCD(23),ORSAMP),
     %(RLHDCD(25),FMTCD),(RLHDCD(27),CDPNO),(RLHDCD(29),SORTCD),
     %(RLHDCD(31),VERTSM),(RLHDCD(51),BINGAN),(RLHDCD(53),AMPREC),
     %(RLHDCD(55),UNITNO),(RLHDCD(57),POLARY)
      DATA RLHDBL/'C',' ','1',' ','C','L','I','E','N','T',24*
     %' ','C','O','M','P','A','N','Y',23*' ',
     %'C','R','E','W',' ','N','O',
     %9*' ','C',' ','2',' ','L','I','N','E',12*' ','A',
     %'R','E','A',24*' ','M','A','P',' ','I','D',26*' ',
     %'C',' ','3',' ','R','E','E','L',' ','N','O',11*' ',
     %'D','A','Y','-',
     %'S','T','A','R','T',' ','O','F',' ','R','E','E','L',5*' ',
     %'Y','E','A','R',
     %6*' ','O','B','S','E','R','V','E','R',18*' ',
     %'C',' ','4',' ','I','N','S','T','R','U','M','E','N','T',':',
     %' ','M','F','G',
     %12*' ','M','O','D','E','L',12*' ','S','E','R','I','A','L',
     %' ','N','O',23*' ',
     %'C',' ','5',' ','D','A','T','A',' ','T','R','A','C','E','S','/',
     %'R','E','C','O','R','D',8*' ',
     %'A','U','X','I','L','I','A','R','Y',' ','T','R','A','C','E',
     %'S','/','R','E','C','O','R','D',9*' ',
     %'C','D','P',' ','F','O','L','D',10*' ',
     %'C',' ','6',' ','S','A','M','P','L','E',' ',
     %'I','N','T','E','R','V','A','L',9*' ',
     %'S','A','M','P','L','E','S','/','T','R','A','C','E',7*' ',
     %'B','I','T','S','/','I','N',6*' ',
     %'B','Y','T','E','S','/','S','A','M','P','L','E',7*' ',
     %'C',' ','7',' ','R','E','C','O','R','D','I','N','G',' ',
```

```
%'F','O','R','M','A','T',8• ' ',
%'F','O','R','M','A','T',' ','T','H','I','S',' ',
%'R','E','E','L',8• ' ',
%'M','E','A','S','U','R','E','M','E','N','T',' ',
%'S','Y','S','T','E','M',10• ' ',
%'C',' ','8',' ','S','A','M','P','L','E',' ','C','O','D','E',':',
%' ','F','L','O','A','T','I','N','G',' ','P','T',5• ' ',
%'F','I','X','E','D',' ','P','T',5• ' ',
%'F','I','X','E','D',' ','P','T','-','G','A','I','N',5• ' ',
%'C','O','R','R','E','L','A','T','E','D',6• ' ',
%'C',' ','9',' ','G','A','I','N',' ',' ','T','Y','P','E',
%':',' ','F','I','X','E','D',5• ' ',
%'B','I','N','A','R','Y',5• ' ',
%'F','L','O','A','T','I','N','G',' ','P','O','I','N','T',5• ' ',
%'O','T','H','E','R',19• ' ',
%'C','1','0',' ','F','I','L','T','E','R','S',':',' ',
%'A','L','I','A','S',5• ' ',
%'H','Z',' ',' ','N','O','T','C','H',5• ' ','H','Z',
%' ',' ','B','A','N','D',5• ' ','-',5• ' ','H','Z',
%' ',' ','S','L','O','P','E',4• ' ','-',4• ' ','D','B','/',
%'O','C','T',
%'C','1','1',' ','S','O','U','R','C','E',':',' ',
%'T','Y','P','E',12• ' ',
%'N','U','M','B','E','R','/','P','O','I','N','T',8• ' ',
%'P','O','I','N','T',' ','I','N','T','E','R','V','A','L',18• ' ',
%'C','1','2',5• ' ','P','A','T','T','E','R','N',':',27• ' ',
%'L','E','N','G','T','H',8• ' ',
%'W','I','D','T','H',18• ' ',
%'C','1','3',' ','S','W','E','E','P',':',' ','S','T','A',
%'R','T',5• ' ','H','Z',
%' ',' ','E','N','D',5• ' ','H','Z',
%' ',' ','L','E','N','G','T','H',6• ' ','M','S',
%' ',' ','C','H','A','N','N','E','L',' ','N','O',5• ' ',
%'T','Y','P','E',8• ' ',
%'C','1','4',' ','T','A','P','E','R',':',' ','S','T','A','R','T',
%' ','L','E','N','G','T','H',7• ' ',
%'M','S',' ',' ',' ','E','N','D',' ','L','E','N','G','T','H',7• ' ',
%'M','S',' ',' ','T','Y','P','E',21• ' ',
%'C','1','5',' ','S','P','R','E','A','D',':',' ','O','F','F',
%'S','E','T',8• ' ',
%'M','A','X',' ','D','I','S','T','A','N','C','E',8• ' ',
%'G','R','O','U','P',' ','I','N','T','E','R','V','A','L',20• ' ',
%'C','1','6',' ','G','E','O','P','H','O','N','E','S',
%':',' ','P','E','R',' ','G','R','O','U','P',5• ' ',
%'S','P','A','C','I','N','G',5• ' ',
%'F','R','E','Q','U','E','N','C','Y',5• ' ',
%'M','F','G',10• ' ','M','O','D','E','L',7• ' ',
%'C','1','7',5• ' ','P','A','T','T','E','R','N',':',
%27• ' ','L','E','N','G','T','H',8• ' ',
%'W','I','D','T','H',18• ' ',
%'C','1','8',' ','T','R','A','C','E','S',' ',
%'S','O','R','T','E','D',' ','B','Y',':',' ',
%'R','E','C','O','R','D',5• ' ',
%'C','D','P',5• ' ',
%'O','T','H','E','R',34• ' ',
%'C','1','9',' ','A','M','P','L','I','T','U','D','E',' ',
%'R','E','C','O','V','E','R','Y',':',' ',
%'N','O','N','E',6• ' ',
%'S','P','H','E','R','I','C','A','L',' ','D','I','V',7• ' ',
%'A','G','C',4• ' ',
%'O','T','H','E','R',14• ' ',
%'C','2','0',' ','M','A','P',' ',
%'P','R','O','J','E','C','T','I','O','N',22• ' ',
%'Z','O','N','E',' ','I','D',7• ' ',
%'C','O','O','R','D','I','N','A','T','E',' ',
%'U','N','I','T','S',10• ' ',
%'C','2','1',' ','P','R','O','C','E','S','S','I','N','G',
%':',65• ' ',
%'C','2','2',' ','P','R','O','C','E','S','S','I','N','G',
%':',65• ' ',
%'C','2','3',77• ' ','C','2','4',77• ' ','C','2','5',77• ' ',
%'C','2','6',77• ' ','C','2','7',77• ' ','C','2','8',77• ' ',
```

```
     %'C','2','9',77*' ','C','3','0',77*' ','C','3','1',77*' ',
     %'C','3','2',77*' ','C','3','3',77*' ','C','3','4',77*' ',
     %'C','3','5',77*' ','C','3','6',77*' ','C','3','7',77*' ',
     %'C','3','8',77*' ','C','3','9',77*' ',
     %'C','4','0',' ','E','N','D',' ',
     %'A','S','C','I','I',67*' '/
      DATA PROC1/'D','E','M','U','L','T','I','P','L','E','X','E','D',
     %51*' '/
      DATA CLIENT/' ','D','U','R','H','A','M',' ','U','N',
     %'I','V','E','R','S','I','T','Y',6*' '/
      DATA CDPFLD/' ','1',8*' '/
      DATA SMPINT/' ','4','0','0','0',4*' '/
      DATA BITS/' ','1','6','0','0',' '/
      DATA BYTES/' ','4',5*' '/
      DATA RECFMT/' ','P','E',5*' '/
      DATA RLFMT/' ','P','E',5*' '/
      DATA UNITS/' ','M','E','T','R','E','S',3*' '/
      DATA SFLTPT/' ','Y','E','S',' '/
      DATA GFLTPT/' ','Y','E','S',' '/
      DATA SOURCE/' ','A','I','R','-','G','U','N','S',3*' '/
      DATA SRCPTN/' ','L','I','N','E','A','R',20*' '/
      DATA GRPINT/' ','1','0','0',16*' '/
      DATA GEOGPN/' ','5','0',' ',' '/
      DATA MTNAM(1)/3RTM4/,MTNAM(2)/3RTM5/,MTNAM(3)/3RTM6/
C
C     ASSIGN CHANNEL NUMBERS
C
 210  ICHN=IDRV+10
C
C     LOOKUP TAPE DRIVE
C
      TYPE 2000,IDRV,IFNUM
      WRITE(1,2000) IDRV,IFNUM
 2000 FORMAT(' START NEW WRITE DRIVE:',I1,' FILE NUMBER:'I4)
      IEOT=0
      ILK=LOOKUP(ICHN,MTNAM(IDRV+1),-1)
      IER=IPEEK("172520)
      ISTUS=IER.AND."100
      IF(ISTUS.GT.0) GOTO 240
      TYPE 2100,IDRV,IFNUM
 2100 FORMAT(' GIVEN DRIVE NO:'I1,' NOT AVAILABLE AT FILE NO:',I4)
      PAUSE
      GOTO 210
C
C     CHECK WRITE DRIVE IS NOT WRITE PROTECTED
C
 240  ISTUS=IER.AND."4
      IF(ISTUS.EQ.0) GOTO 220
      TYPE 2900,IDRV,IFNUM
 2900 FORMAT(' GIVEN WRITE DRIVE NO:',I1,' IS WRITE PROTECTED')
      PAUSE
      GO TO 210
C
C WRITE TAPE HEADER BLOCKS IF AT START OF TAPE
C
 220  ISTUS=IER.AND."40
      IF(ISTUS.EQ.0) RETURN
      DO 100 I=1,23
 100  AREA(I)=AREANO(I)
      DO 130 I=1,4
 130  DAY(I)=DAYNO(I)
      DO 110 I=1,5
 110  YEAR(I)=YEARNO(I)
      DO 120 I=1,17
 120  OBSVER(I)=OBSVNO(I)
      ENCODE(12,1010,LINE,ERR=30) LINENO
      ENCODE(11,1010,REELNO,ERR=30) REELNU
      ENCODE(8,1010,TRINRC,ERR=30) NCHAN
      ENCODE(7,1010,SPINTR,ERR=30) NSAMP
 1010 FORMAT(I5)
      ENCODE(8,1010,SORCNO,ERR=30) IGUNNO
      ENCODE(18,1010,SCPTIN,ERR=30) SLSPAC
```

```
      ENCODE(18,1010,SWIDTH,ERR=30) IWIDTH
      ENCODE(8,1010,SPDOFS,ERR=30) SROFF
      ENCODE(8,1010,SPDDIS,ERR=30) ITOTAL
      GO TO 40
 30   STOP 'ENCODE ERROR'
 40   IF(IWMTLM(RLHDBL(1,1),ICHN,1600,ITMERR(1)).NE.1600) STOP 'ERROR
     + WRITING ASCII TAPE HEADER'
      CALL ISPFNW("377,ICHN)
      DO 101 M=1,400
101   RLHDCD(M)=0
      TRCNUM=NCHAN
      AUXTRS=0
      SAMPIN=4000
      ORSINT=4000
      FMTCD=1
      CDPNO=1
      SORTCD=1
      VERTSM=1
      BINGAN=1
      AMPREC=1
      UNITNO=1
      POLARY=3
      NOSMPL=NSAMP
      ORSAMP=NSAMP
      TRCNUM=NCHAN
      CALL JICVT(LINENO,LNNOL)
      CALL JICVT(REELNU,RLNOL)
      CALL JJCVT(LNNOL)
      CALL JJCVT(RLNOL)
      IF(IWMTLM(RLHDCD(1),ICHN,200,ITMERR(1)).NE.200) STOP 'ERROR
     + WRITING BINARY TAPE HEADER'
      CALL ISPFNW("377,ICHN)
      RETURN
      END




      SUBROUTINE FRAMFL(BUFF,FRAME,IST,IFIN,ICODE)
      VIRTUAL BUFF(8448)
      INTEGER*2 FRAME(33),SPOS,BUFEND,BUFF,GAINS(3840),GSAVE(30),
     %BSST1,BLST,OLAP,RST,IWORD,IWORDB,IWORDF,DISKAD
      LOGICAL*1 IF(2),IBYT(2),IBYTE(2),OUTBUF(16624),RC
      COMMON/DATBUF/ OUTBUF
      COMMON /SUBS/GSAVE,IFDIR,IERR,IERFLG,SPOS,RC
      COMMON/BUFCOM/BSST1,BLST,OLAP,RST,DISKAD,ITSIZ
      EQUIVALENCE (IWORD,IF(1)),(IWORDB,IBYTE(1)),(IWORDF,IBYT(1))
      EQUIVALENCE (GAINS(1),OUTBUF(241))
      IF(ICODE.GE.0)GOTO 1
      BUFEND=8449
      IBEG=4097
      RETURN
C
C FILL FRAME IN NORMAL CIRCUMSTANCES
C
    1 IF(ICODE.GT.1)GOTO 10
      IF(ICODE.EQ.1)FRAME(IST)=IWORDF
      IF(ICODE.EQ.1)IST=IST+1
      DO 20 I=IST,IFIN
      FRAME(I)=IBTSWP(BUFF(SPOS))
      SPOS=SPOS+1
      IF(SPOS.LT.BUFEND)GOTO 20
      IBEG=4096-IBEG+2
      IFILL=1
      IF(IBEG.EQ.4097)IFILL=4097
      CALL DISKRD(BUFF(IFILL),4096,DISKAD,NERR)
      IF(NERR.LT.0) STOP 'FRAM READ ERR'
      SPOS=IBEG
      BUFEND=SPOS+4096
   20 CONTINUE
      RETURN
```

```
C
C BYTE LOST PATTERN FRAME FILL
C
   10 IF(ICODE.EQ.2)IWORDF=FRAME(33)
      DO 40 I=IST,IFIN
      IWORDB=IBTSWP(BUFF(SPOS))
      IF(2)=IBYT(1)
      IF(1)=IBYTE(2)
      FRAME(I)=IWORD
      IWORDF=IWORDB
      SPOS=SPOS+1
      IF(SPOS.LT.BUFEND)GOTO 40
      IBEG=4096-IBEG+2
      IFILL=1
      IF(IBEG.EQ.4097)IFILL=4097
      CALL DISKRD(BUFF(IFILL),4096,DISKAD,NERR)
      IF(NERR.LT.0) STOP 'FRAM READ ERR'
      SPOS=IBEG
      BUFEND=SPOS+4096
   40 CONTINUE
      RETURN
      END



      SUBROUTINE DMXOUT(IBLKOF,IDRV,NCHNO,NWRD,FTRCNO,IFNUM,IEOT)
C
C NCHAN NUMBER OF OUTPUT TRACES PER FILE
C
C IBLKOF POSITIONS OF OUTPUT TRACES IN DISC FILE
C
C OUTBUF DATA TRANSFER ARRAY
C
C IDRV WRITE DRIVE NUMBER
C
C NCHNO IS THE OUTPUT R.T.11 CHANNEL FOR DISC READS
C
C FTRCNO CURRENT TRACE NUMBER
C
C NWRD NUMBER OF DATA WORDS PER TRACE (2*NSAMP)
C
C IFNUM CURRENT FILE NUMBER
C
C IEOT END OF TAPE FLAG
C
      INTEGER*4 JCHAN,JTRCNO
      INTEGER*2 IBLKOF(30),CHOFF(30),NBLKOF(30),ICHAN(30),GCNT,ITMERR(2)
      LOGICAL*1 OUTBUF(16624)
      COMMON/DATBUF/ OUTBUF
      COMMON/DECOM/ CHOFF,ICHAN,NCHAN,NBLKOF,GCNT,IHSTRT,IHSEC
      EQUIVALENCE (JCHAN,OUTBUF(13)),(JTRCNO,OUTBUF(1)),
     $(IDUM,OUTBUF(189))

      ITRY=0
      NWORD=NWRD+120
C
C     ASSIGN CHANNEL NUMBERS
C
      ICHN=IDRV+10
C
C TRANFER DATA IN BLOCKS OF ONE CHANNEL PLUS H/BLK
C
      DO 200 I=1,NCHAN
C
C WRITE CHANNEL NUMBER IN HEADER BLOCK
C
      IICHAN=I
      IDUM=ICHAN(I)
      CALL JICVT(IICHAN,JCHAN)
      CALL JJCVT(JCHAN)
```

```
      CALL JAFIX(FTRCNO,JTRCNO)
      CALL JJCVT(JTRCNO)
      IBLK=IBLKOF(1)
      IF(IREADW(NWRD,OUTBUF(241),IBLK,NCHNO).LT.0) STOP 'TRACE READ ERR'
 220  IWT=IWMTLM(OUTBUF(1),ICHN,NWORD,ITMERR(1))
C
C CHECK FOR ERRORS
C
      IF(IWT.EQ.NWORD) GOTO 210
      IF(IWT.EQ.-1) GOTO 400
C
C RE-TRY
C
  240 ITRY=ITRY+1
      CALL ISPFNW("375,ICHN)
      CALL ISPFNW("376,ICHN)
      WRITE(1,2200) IDRV,IFNUM,I,ITMERR(1)
      TYPE 2200,IDRV,IFNUM,I,ITMERR(1)
 2200 FORMAT(' ERROR ON DRIVE:',I2,' FILE NO:',I4,' CHN NO:',I4,
     $' STATUS:',O6)
      IF(ITRY.LT.3) GO TO 220
C
C BAD OUTPUT FILE SO REWIND TO LAST FILE AND CLOSE TAPE
C
      CALL BACKSP(IICHAN,ICHN)
      CALL ISPFNW("377,ICHN)
      IFNUM=IFNUM-1
      TYPE 2400,IFNUM,IDRV
      WRITE(1,2400) IFNUM,IDRV
 2400 FORMAT(' FILE NO:',I4,' IS LAST WRITTEN ON DRIVE:',I2)
      STOP 'BAD O/P TAPE - REMEMBER TO REWIND READ TAPE 1 FILE'
C
C COME HERE TO WRITE TAPE MARK
C
 210  CALL ISPFNW("377,ICHN)
      ISTUS=IPEEK("172520)
      IF((ISTUS.AND."40000).LE.0) GO TO 240
      IF((ISTUS.AND."2000).EQ.0) GOTO 200
      CALL ISPFNW("375,ICHN)
      GOTO 400
 200  FTRCNO=FTRCNO+1.0
C
C NORMAL RETURN
C
      RETURN
C
C COME HERE FOR EOT
C
 400  IEOT=1
      CALL BACKSP(IICHAN,ICHN)
      FTRCNO=FTRCNO-FLOAT(IICHAN)+1.0
C
C CLOSE REWIND OUTPUT TAPE
C
      CALL ISPFNW("377,ICHN)
      CALL ISPFN ("372,ICHN)
      CALL CLOSEC(ICHN)
      TYPE 2300,IDRV,(IFNUM-1)
      WRITE(1,2300) IDRV,(IFNUM-1)
 2300 FORMAT(' EOT ON DRIVE:',I2,' FILE NO:',I4,/,
     $' GIVE NEW WRITE DRIVE NUMBER:',$)
      ACCEPT 1000,IDRV
 1000 FORMAT(I1)
      IF(IDRV.GT.2) STOP 'WRITE TERMINATION'
      RETURN
      END
```

```
; EXTENDED TAPE FILE READ SUBROUTINE - DSTST
  .SBTTL INTRO
;
; ENABLES APPROX. FIRST 14 SECS OF ANY LENGTH RECORD (FILE)
; TO BE READ FROM A FIELD TAPE USING STANDARD TAPE DRIVES
; AND A TM11 LIKE CONTROLLER ON A PDP 11 MACHINE WITH AT LEAST 12BK WORDS
; OF MEMORY. INTENDED FOR USE UNDER RT 11 V03 OR LATER WITH CERTAIN
; RESTRICTIONS ON ITS USE AND SOME UNPLEASANT SIDE EFFECTS WHICH MAY OR
; MAY NOT AFFECT USER APPLICATIONS.
;
;
; REQUIRES FORMATTED DISC RK5
;
;

  .PAGE
  .PSECT DSTST


 .MCALL .PRINT,.EXIT

;*****************************************************************
; SECTION TO INTERFACE TO RT 11 FORTRAN IV
;*****************************************************************

; CALLING            EXREAD(COMMAND,DRIVE,STATUS,IFNUM,IEOTR,ILEN)
;
; WHERE       COMMAND      IS AN INTEGER CONTROLLING THE FUNCTION
;      TO BE EXECUTED
;
;      DRIVE       IS THE DRIVE NUMBER (0-2)
;
;      STATUS      IS THE STATUS OF THE DRIVE OR THE READ
;             RETURNED BY THE SUBROUTINE
;
;      IFNUM        IS THE NUMBER OF THE FILE ON TAPE
;
;      IEOTR        IS A FLAG SET TO INDICATE EOT
;
;                ILEN IS READ COUNTER = NO. OF 4096 WORD BLKS READ
;
; DECLARATIONS

FNUM:   .WORD 1    ;CURRENT FILE POSITION
SAMRAT:   .WORD 250.    ;SAMPLES PER  SEC
DRIVE:   .WORD 1    ;CURRENT DRIVE NUMBER


  .SBTTL DECLARATIONS



  IDERR  = -9.  ;ID ERROR ON DISC
  STERR  = -8.  ;READ TAPE NOT AVAILABLE
  SKERR  = -3 ;SEEK ERROR
  MEMERR  = -4 ;OVERLAY WRITE ERROR
  RERR  = -5 ;FILE READ ERROR
  ILCOM  = -1 ;ILLEGAL COMMAND ERR
  SPCERR  = -2 ;ERROR IN SPACING
  REV  = 60013 ;SPACE BACK
  FOR  = 60011 ;SPACE FORWARD

  MTS = 172520
  RKDS = 177400
  XCSR = 177564
  RCSR = 177560
  CLK = 177546    ;LTC STATUS REG

  WRITE = 3
  READ = 5
  SEEK = 11
```

```
  DISCNT = ^C3072.

  DISCAD = 120001
  FILAD = 122000



  .SBTTL INTERFACE
  .PAGE

EXREAD:: MOV @#177640,-(SP)
  CLR @#177640
  MOV R0,-(SP)   ;SAVE REGISTERS
  MOV R1,-(SP)
  MOV R2,-(SP)
  MOV R3,-(SP)
  MOV R4,-(SP)


  TST (R5)+    ;BUMP R5 TO POINT AT VARIABLE POINTERS
  CALL IDCHK    ;CHECK PLATTER VALID
  MOV @2(R5),DRIVE  ;DRIVE
  MOV @(R5),R0   ;FUNCTION CODE
  TST R0    ;DECODE FUNCTION
  BLE 3$
  JMP RFIL   ;READ THE FILE
3$:  MOV @6(R5),R1 ;NUMBER OF FILES
  BGT FORW   ;FORWARD SPACE
  BLT BACK   ;BACKWARDS
  JMP EXIT



  .PAGE
  .SBTTL SPACING FUNCTIONS



BACK:  ASL R1   ;DOUBLE TO ALLOW FOR TMS
  MOV #MTS,R2   ;BASE REGISTER FOR TAPE
  MOV R1,4(R2) ;LOAD COUNT FOR BLOCKS TO SPACE
  MOV DRIVE,R3 ;BUILD COMMAND WORD
  ASH #8.,R3   ;DRIVE
  MOV R3,2(R2) ;SELECT DRIVE
  BIS #REV,R3   ;COMMAND

  CALL WAIT   ;WAIT FOR TAPE READY
1$:  MOV R3,2(R2) ;EXECUTE
  CALL WAIT
  BIT #40,(R2) ;BEGINNING OF TAPE MARK?
  BLE 4$   ;BOT NOT SET
  INC 4(R2)   ;INCREMENT RECORD COUNTER
  TST 4(R2)   ;ANY MORE FILES TO REWIND?
  BGE 3$   ;NO MORE = NORMAL EXIT
  BR ERR5   ;SPACING ERROR
4$:  TST 4(R2)   ;ANY MORE FILES TO REWIND?
  BLT 1$   ;LOOP TILL ALL DONE
3$:  TST R0   ;CHECK THE COMMAND
  BEQ EXIT   ;EXIT IF THAT IS ALL
  JMP RFIL   ;READ THE FILE



FORW:  NEG R1   ;MAKE COUNT NEGATIVE
  ASL R1   ;DOUBLE IT
  MOV #MTS,R2   ;BASE ADDRESS OF TAPES
  MOV R1,4(R2) ;LOAD COUNT
  MOV DRIVE,R3 ;BUILD COMMAND WORD
```

```
  ASH #8.,R3  ;DRIVE
  MOV R3,2(R2) ;SELECT DRIVE
  BIS #FOR,R3  ;COMMAND
  CALL WAIT  ;ARE WE READY?
2$:  MOV R3,2(R2) ;EXECUTE COMMAND
  CALL WAIT  ;ER ARE WE READY AGAIN?
  BIT #2000,(R2) ;HAVE WE REACHED EOT?
  BGT ERR5  ;YES SO ERROR
  TST 4(R2)  ;ANY MORE TO GO?
  BLT 2$  ;AND AROUND WE GO AGAIN
  TST R0  ;CHECK THE COMMAND
  BEQ EXIT  ;EXIT IF THAT IS ALL
  JMP RFIL  ;AND READ THE BUGGER



;*****************************************************************
; SUBROUTINE TO WAIT FOR THE TAPE DRIVES TO BE READY
;*****************************************************************


WAIT:  TSTB 2(R2)  ;IS CONTROLLER READY?
  BPL WAIT  ;NO THE CONTROLLER IS STILL ASLEEP
       BIT #1,(R2)  ;IS DRIVE READY?
  BEQ WAIT  ;NOT YET SO WAIT
  RTS PC  ;ALL READY SO RETURN

;NOTE THAT THIS WAIT ROUTINE ASSUMES MTS IN R2 AND DOES NOT CHECK FOR ERRORS

;DRIVE: .WORD 1  ;CURRENT DRIVE NUMBER


  .PAGE
  .SBTTL MISC SUBROUTINES



;*************************************************************************
; CHECK DRIVE VALID SUBROUTINE.....REQUIRES THAT DISC IS FORMATTED
;*************************************************************************

IDCHK: MOV #RKDS,R1  ;BASE REG
 MOV #BUFFER,10(R1)  ;BUFFER
 MOV #-256.,6(R1)  ;ONE BLOCK
 MOV #DISCAD,R0  ;DISCADDRESS
 DEC R0  ;PREVIOUS BLOCK
 MOV R0,12(R1)  ;LOAD IT
 MOV #READ,4(R1)  ;READ IT
2$: TSTB 4(R1)  ;DONE OR NOT
 BPL 2$  ;NOT YET
 TST 4(R1)  ;ALL OK?
 BPL 3$  ;YES
 JMP ERR10
3$: CLR R1  ;SET UP COUNTER
 MOV #10.,R0  ;AND POINTER
4$: CMP BUFFER(R1),ID(R1)
 BNE ERR10  ;DISC ID ERR
 TST (R1)+  ;BUMP COUNT+POINTER
 DEC R0
 BGT 4$  ;AROUND WE GO
 RETURN



ID: .ASCIZ /EXREAD SCRATCH DISK /
 .EVEN
```

```
; ERROR HANDLING ROUTINES
ERR5:  MOV #SPCERR,@4(R5)  ;SPACE ERR
  CLR @10(R5)    ;CLEAR EOT
  BIT #2000,(R2)  ;IS IT EOT?
  BEQ 1$    ;NOPE
  MOV #1,@10(R5)  ;FLAG IT
1$:  JMP EREXIT  ;ERROR EXIT

ERR6:  MOV #MTS,R2  ;BASE
  MOV #ILCOM,4(R2)  ;ILLEGAL COMMAND
  JMP EREXIT


ERR10:  MOV #IDERR,@4(R5)  ;FLAG ERROR
  TST (SP)+  ;POP CALL
  JMP EREXIT
```

```
;*********************************************************************
; NORMAL EXIT POINT AFTER SUCCESSFUL READ AND REFORMATTING
;*********************************************************************

EXIT:  CLR @4(R5)  ;CLEAR THE STATUS FLAG
```

```
;*********************************************************************
; ERROR ROUTINE EXIT SECTION....ALLOWS ERRORS TO ESCAPE TO CALL
;*********************************************************************


EREXIT:  MOV (SP)+,R4
  MOV (SP)+,R3
  MOV (SP)+,R2
  MOV (SP)+,R1
  MOV (SP)+,R0
  MOV (SP)+,@#177640

  TST -(R5)
  RTS PC  ;RETURN TO FORTRAN

;*********************************************************************
```

```
;*********************************************************************
; ENTRY POINT AT CALL
;*********************************************************************

RFIL: MOV #RELOC,RELOC  ;SET RELOCATION POINTERS


; DISABLE ANY INTERUPTS THAT COULD OCCUR

 BIC #100,@#CLK  ;LTC DISABLE
```

```
 BIC #100,@#XCSR  ;TERMINAL
 BIC #100,@#RCSR
 MOV #RKDS,R1  ;BASE ADDRESS
 BIC #100,4(R1)  ;DISC INTERUPTS

; WRITE MEMORY TO DISC

 BIC #60,4(R1)  ;CLEAR EXTENDED MEM ADD BITS
 CLR 10(R1)   ;START TRANSFER FROM BOTTOM
 MOV #DISCAD,12(R1)  ;START OF DISC

2$: TST 4(R1)  ;ANY ERROR?
 BPL 1$   ;NONE SO CONTINUE
 JMP ERR1   ;ERROR ON SEEK
1$: TSTB 4(R1)   ;ARE WE READY?
 BPL 2$   ;NOT YET

 MOV #-5120.,6(R1)  ;ALL OF MEMORY
 MOV #WRITE,4(R1)  ;START WRITE
4$: TST 4(R1)   ;ANY ERRORS?
 BPL 3$   ;ALL OK
 RESET
 JMP RFIL   ;ERROR ON WRITE
3$: TSTB 4(R1)  ;ARE WE READY?
 BPL 4$   ;NOT YET
; WAIT FOR MEMORY EXTENDED BITS TO OVER FLOW
 BIT #20,4(R1)  ;OVERFLOW?
 BEQ 1$   ;NOT YET SO LOOP

; NOW SET UP A NEW PROGRAM STACK READY FOR RELOCATED PROGRAM

 MOV #1000,R0  ;R0 IS STACK POINTER
 MOV SP,-(R0)  ;OLD STACK IS FIRST ENTRY
 MOV R5,-(R0)  ;SAVE ALL REGISTERS
 MOV R4,-(R0)
 MOV R3,-(R0)
 MOV R2,-(R0)
 MOV R1,-(R0)


; RELOCATE PROGRAM TO BOTTOM OF MEMORY

 MOV RELOC,R2  ;FROM
 CLR R4   ;TO
 MOV SIZE,R3   ;NO OF WORDS TO RELOCATE

5$: MOV (R2)+,(R4)+  ;DO TRANSFER
 DEC R3   ;BUMP COUNT
 BGT 5$   ;STILL SOME MORE

;NOW GET READ DRIVE NUMBER

 MOV DRIVE,R4  ;DRIVE NUMBER
 ASH #8.,R4   ;SELECT DRIVE
 BIS #60002,R4  ;READ COMMAND
 CLR R5   ;CLEAR EOT FLAG

 MOV @#6,R3  ;RELOCATED ENTRY POINT
 JMP (R3)   ;SO REENTER AND PRAY



  .PAGE
  .SBTTL RELOCTBL. CODE
;**********************************************************
; EXECUTABLE RELOCATED CODE
;**********************************************************
; VARIABLES USED IN RELOCATION AND RESTARTS


RELOC: .WORD 0
```

```
SIZE: .WORD <TIDY-RELOC>/2  ;SIZE OF PROGRAM IN WORDS
OFFSET: .WORD <REENT-RELOC>  ;REENTRY POINT OFFSET
STOFF: .WORD <STENT-RELOC>  ;RELOCATED ENTRY POINT OFFSET


STENT: RESET    ;DUMMY INSTRUCTION

; SET UP TAPE READ FROM SELECTED DRIVE

 MOV #1,SP    ;INITIALIZE READ BLK COUNTER
 MOV #MTS,R1    ;BASE ADDRESS OF REGISTERS
 BIC #1,R4    ;CLEAR GO BIT FOR TM RE-TRY
 MOV R4,2(R1)   ;LOAD SELECT BITS &COMMAND
 TST 2(R1)    ;ALL OK
 BMI ERRSET    ;NO SO ERROR
 BIT #1,(R1)    ;TAPE DRIVE READY?
 BEQ ERRSET    ;IF NOT THEN BOMB OUT
 MOV #1000,6(R1)   ;SET UP LARGE BUFFER
 BIC #60,2(R1)   ;IN FIRST 32K WORDS
 MOV #-9192.,4(R1)   ;AND BYTE COUNT
 BIS #1,R4    ;LOAD GO BIT
 MOV R4,2(R1)   ;START THE READ

; TIMING IS NOW CRITICAL


1$: TST 2(R1)    ;CHECK STATUS OF TAPE
 BMI EOF    ;ERROR AS EXPECTED
 BIT #1,(R1)    ;IS DRIVE STILL BUSY?
 BNE EOF    ;ASSUME DONE IF NOT BUSY
 MOV 4(R1),R3   ;LOOK AT BYTE COUNTER
 ADD #1000.,R3   ;TIME TO TOP UP?
 BLT 1$    ;BRANCH IF BYTE COUNT IS MORE THAN -1000
 MOV #-9192.,4(R1)   ;TOP UP BYTE COUNT
 INC SP    ;BUMP READ BLK COUNTER
 BR 1$    ;BACK WE GO

; ERROR CHECK THE CAUSE.......EXPECT EOF OR NEX

EOF: MOV #100.,R3   ;TIME OUT COUNT
2$: DEC R3    ;BUMP IT
 BLT ERRSUP    ;IF CONTROLLER DOESN'T COME READY
 BIT #200,2(R1)   ;CHECK CONTROLLER READY
 BEQ 2$    ;NOT READY YET
 BIT #40000,(R1)   ;IS IT AN EOF?
 BEQ 7$    ;NO EOF SET SO BRANCH
 INC R5    ;BUMP LEOT FLAG
 CMP #1,R5    ;IS IT AN LEOT?
 BEQ STENT    ;NO LEOT, SO READ NEXT FILE
 MOV #-1,-(R0)   ;SET LEOT FLAG
 MOV #0,-(R0)   ;SET NO ERRORS
 BR RECVR
7$: BIT #2000,(R1)   ;IS EOT SET?
 BEQ 3$    ;NO SO PUSH OK ON STACK
 MOV #1,-(R0)   ;YES SO FLAG IT
 JMP DRED    ;EOF BUT DATA STILL VALID
3$: MOV #0,-(R0)   ;OK
4$: BIT #100000,2(R1)   ;IS IT AN ERROR EOF?
 BEQ DRED    ;NO SO WRITE DATA
 BIT #200,(R1)   ;IS IT A NEX?
 BNE DRED    ;YES
 BR ERR    ;UNEXPECTED ERROR CONDITION

ERRSUP: MOV #0,-(R0)   ;PUSH EOT FLAG
ERR: NEG SP    ;ERROR READ BLK RETURN
 MOV SP,-(R0)   ;PUSH READ BLK COUNT ON STACK
 BR RECVR    ;RECOVER RT11

ERRSET: MOV #0,-(R0)   ;PUSH EOT FLAG
 MOV #0,-(R0)   ;PUSH MAGTAPE NOT AVAILABLE FLAG
 BR RECVR
```

```
;-------------------------------------------------------------------------

   .PAGE
   .SBTTL WRITE DATA

;**********************************************************************
; SEEK FILE POSITION AND THEN  WRITE DATA TO DISC
;**********************************************************************

DRED: MOV #RKDS,R1   ;BASE REGS FOR DISC

2$: TST 4(R1)    ;ALL OK?
 BPL 3$    ;NO
 RESET     ;RESET AND TRY AGAIN
 BR DRED
3$: TSTB 4(R1)    ;ARE WE READY?
 BPL 2$    ;NOT YET

; NOW WRITE DATA TO DISC

 MOV #FILAD,12(R1)   ;DISC ADDRESS OF FILE
 BIC #60,4(R1)   ;AND EXTENDED ADDRESS BITS
 MOV #1000,10(R1)   ;TRANSFER FROM BEG OF BUFFER
 MOV #48.,R2    ;SET UP LOOP COUNT
8$: MOV #-2560.,6(R1)   ;WORD COUNT
 BIS #WRITE,4(R1)

9$: TST 4(R1)    ;ANY ERRORS?
 BPL 7$    ;NO CONTINUE
 BR ERR    ;ERROR
7$: TSTB 4(R1)   ;ARE WE EADY
 BPL 9$    ;NOT YET
 DEC R2    ;LOOP CONTROL
 BGT 8$
 MOV SP,-(R0)   ;PUSH READ BLK COUNT ON STACK


; OK DONE NOW RELOCATE AGAIN



RECVR: NOP
 MOV @#0,R2    ;RELOACTE TO
 MOV #0,R1    ;FROM
 MOV @#2,R3    ;SIZE IN WORDS

1$: MOV (R1)+,(R2)+   ;TRANSFER
 DEC R3    ;MORE
 BGT 1$    ;YES

; ALL DONE RECALCULATE ENTRY POINT

 MOV @#0,R1    ;BASE ADDRESS
 ADD @#4,R1   ;ADD OFFSET
 JMP (R1)    ;OFF WE GO AGAIN... I HOPE

; RE ENTRY MODULE

REENT: MOV (R0)+,R1   ;STATUS FLAG
 MOV (R0)+,R2   ;EOT FLAG POPPED
 TST (R0)+    ;R1 NOT REQUIRED
 TST (R0)+    ;NEITHER IS R2
 MOV (R0)+,R3
 MOV (R0)+,R4
 MOV (R0)+,R5
 MOV (R0)+,SP   ;ORIGINAL STACK
 MOV R1,R0    ;R0 IS STATUS NOW
 MOV #RKDS,R1   ;BASE REG FOR DISC
7$: MOV #DISCAD,12(R1)   ;MEMORY IMAGE ADDRESS
 BIC #60,4(R1)   ;AND THE WINE LIST WAITER PLEASE
 CLR 10(R1)    ;START AT BOTTOM
```

```
1$: TST 4(R1)    ;CHECK STATUS
 BPL 2$    ;OK
 RESET
 JMP 7$
2$: TSTB 4(R1)    ;ARE WE READY?
 BPL 1$   ;NO NOT YET

 MOV #-5120.,6(R1)   ;ALL OF MEMORY
 MOV #READ,4(R1)   ;READ IT BACK

4$: TST 4(R1)
 BPL 3$
 RESET
 JMP 7$   ;FATAL READ ERROR
3$: TSTB 4(R1)    ;ARE WE READY?
 BPL 4$    ;NOT YET

; NOW CHECK FOR OVERFLOW

 BIT #20,4(R1)   ;EXTENDED MEM ADD BITS
 BEQ 1$

; OK MEMORY RECOVERED SO SHOULD BE OK||||||||



   .PAGE
   .SBTTL TIDYUP



;*****************************************************************
; COMPLETION SECTION..... RESETS ALL DEVICES ON BUS FOR SAFETY
;*****************************************************************

TIDY: RESET
 BIS #100,@#CLK   ;ENABLE CLOCK
 BIS #100,@#XCSR   ;ENABLE TTY
 BIS #100,@#RCSR
 MOV R2,@10(R5)   ;TRANSFER EOT FLAG TO FIV
 TST R2    ;TST EOTR FLAG
 BLT 1$    ;BRANCH IF LEOT DETECTED
 BEQ 2$    ;BRANCH IF EOT NOT DETECTED
 TST R0    ;EOT FOUND - TEST STATUS
 BLT 3$    ;BRANCH IF READ ERROR FOUND
 BR 4$    ;EOT NORMAL RETURN

2$: TST R0    ;NO EOT OR LEOT - TEST STATUS
 BEQ 7$    ;READ DRIVE NOT AVAILABLE ERROR
 BLT 3$    ;BRANCH IF READ ERROR FOUND

4$: DEC R0    ;CORRECT READ BLK COUNTER OFFSET
1$: MOV R0,@12(R5)   ;RETURN READ BLK COUNTER TO FORTRAN
 JMP EXIT

3$: NEG R0    ;EOT READ ERROR - CORRECT READ BLK POLARITY
 DEC R0    ;CORRECT READ BLK COUNTER OFFSET
 MOV R0,@12(R5)   ;RETURN READ BLK COUNT TO FORTRAN
 MOV #RERR,@4(R5)   ;SET STATUS = -5 TO FORTRAN
6$: JMP EREXIT

7$: MOV R0,@12(R5)   ;CLR READ BLK CONTER FOR FORTRAN
 MOV #STERR,@4(R5)   ;SET STATUS = -8 IN FORTRAN
 BR 6$




;*****************************************************************
```

```
;ERROR HANDLING SECTION..... USES PROGRAMMED REQUESTS SO BEWARE|
;*****************************************************************

ERR1: MOV #SKERR,@4(R5)  ;SEEK ERROR
 JMP EREXIT
ERR2: MOV #MEMERR,@4(R5)  ;MEMORY OVERLAY WRITE ERROR
 JMP EREXIT
ERR3: HALT     ;OVERLAY READ ERROR

BUFFER: .BLKW 256.
END: .WORD 0
 .END




 .TITLE DISKRD
;CALLING DISKRD(VARRAY(I),NWRD,DISKAD,IERR)
 .GLOBL DISKRD
 READ = 5
 RKDS = 177400
 USP0 = 177640

DISKRD: MOV #4,R4    ;RE-TRY COUNT
4$: MOV @#USP0,R0  ;GET EXTENSION IN R0
 ASL R0    ;GET XMEM BITS
 ASL R0
 MOV R0,R2    ;INTO R2
 ASL R0    ;GET ADDRESS BITS
 ASL R0
 ASL R0
 ASL R0
 CALL WAIT    ;WAIT FOR DISK CONTROLLER
 BIC #177700,R1  ;CLEAR UNWANTED BITS IN R0
 BIC #77,R0    ;CLEAR UNWANTED BITS IN R1
 BIS R1,R0    ;COMBINE R0 AND R1 TO FORM ADDRESS
 MOV R0,ADDR    ;SAVE START ADDRESS
 MOV R0,10(R3)  ;LOAD INTO DISK ADDR REG.
 SWAB R2    ;MOVE XMEM BITS TO POSITION
 BIC #177717,R2  ;CLEAR UNWANTED BITS

 BIC #100,4(R3)  ;DISABLE INTERUPT

 MOV @4(R5),R0  ;GET WORD COUNT
 NEG R0    ;CONVERT TO 2'S COMP
 MOV R0,6(R3)   ;SET WORD COUNT REG
 MOV 4(R3),R1   ;GET DISK STATUS REG
 BIC #77,R1    ;CLEAR XMEM & FUNC BITS
 BIS R2,R1   ;SET XMEM BITS
 MOV R1,4(R3)   ;LOAD DISK STATUS REG
 MOV @6(R5),12(R3)  ;DISK ADDR TO START READ

1$: TSTB 4(R3)    ;IS DISK READY
 BPL 1$
 BIS #READ,4(R3)  ;COMMENCE READ

2$: TSTB 4(R3)    ;FINISHED?
 BPL 2$
 TST 4(R3)    ;ANY ERRORS?
 BPL 3$    ;BRANCH IF NO ERROR
 DEC R4    ;DECREMENT RETRY COUNTER
 BGE 4$    ;TRY AGAIN
 JMP ERR9    ;PERM READ ERROR

3$: MOV 12(R3),@6(R5)  ;RETURN DISK READ FINISH ADDR
 BIC #60,4(R3)   ;CLEAR XMEM BITS
 CALL WAIT    ;WAIT FOR DISK CONTROLLER
 JMP EXIT

ERR9: MOV #-1,@10(R5)  ;SET ERROR BIT
```

```
EXIT: RTS PC

ADDR: .WORD 0


; SUBROUTINE TO WAIT FOR DISK CONTROLLER

WAIT: MOV #RKDS,R3    ;BASE ADDRESS
 TST 4(R3)
 BPL 1$
 BR WAIT

1$: TSTB 4(R3)
 BPL WAIT

 RTS PC


END: .WORD 0
 .END




 .TITLE IBTSWP

 .GLOBL IBTSWP,IGCHK,IGAIN,IDIRG
IBTSWP: MOV @2(R5),R0
 SWAB R0
 RTS PC

IGCHK: MOV @2(R5),R0
 TSTB R0
 BPL 1$
 MOV #1,@4(R5)
 BR 2$
1$: MOV #-1.,@4(R5)
2$: BIC #177760,R0
 RTS PC

IDIRG: TSTB @2(R5)
 BPL 3$
 MOV #1,R0
 RTS PC
3$: MOV #-1.,R0
 RTS PC

IGAIN: MOV @2(R5),R0
 ASR R0
 BCC 4$
 ADD @6(R5),@4(R5)
4$: TST R0
 BPL 5$
 ADD #1,R0
 BIS #40000,R0
5$: RTS PC

 .END
```

```
 $TITLE DMXA
 $ENTRY DMXA,6
" THIS PROGRAM DOES A FAST DEMUX
" OF SEG A FORMAT DATA PRESENTED AS A 4K
" BLOCK STARTING AT ADDRESS ZERO WITH THE
" PREVIOUS GAINS ELSEWHERE IN MEMORY
" THE GAIN ADDRESS THE DEMUXED OUTPUT ADDRESS AND INCREMENT ARE INPUT
" TO THE ROUTINE ALONG WITH THE NO OF THE GAIN CHECK AND THE
" GAIN CHANGE DIRECTION
 GAIN $EQU 0
 DBASE $EQU 1
 DINC $EQU 2
 GCNT $EQU 3
 DIR $EQU 4
 N1 $EQU 5
 DADR $EQU 6
 DIRCK $EQU 7
 DATA $EQU 7
 IADR $EQU 10
 TEMP $EQU 11
 N $EQU 12
 N26 $EQU 13
 SYNC $EQU 14
 BIAS $EQU 15
 TWO $EQU 16
 MASK $EQU 17
 ERR $EQU 17
"END OF REGISTER ASSIGNMENTS START OF CODE
DMXA: LDSPI IADR;DB=0    "FIRST BYTE SWAP DATA VALUES
 LDSPI N26;DB=1
 LDSPI N;DB=4096.
 MOV IADR,DADR;SETMA
 SUB N26,DADR
 LDSPI SYNC;DB=377
 DEC N;LDDPA;DB=1
LOOP: LDSPI DATA;DB=MD   "GET EACH DATA VALUE INTO DATA
 ADD N26,IADR;SETMA
 MOVL DATA,TEMP    "COPY TO TEMP AND SHIFT LEFT 8 TIMES
 MOVL TEMP,TEMP
 MOVL TEMP,TEMP
 MOVL TEMP,TEMP
 MOVL TEMP,TEMP
 MOVL TEMP,TEMP
 MOVL TEMP,TEMP
 MOVL TEMP,TEMP
 MOVRR DATA,DATA    "SHIFT DATA RIGHT 8 TIMES
 MOVRR DATA,DATA
 MOVRR DATA,DATA
 MOVRR DATA,DATA
 AND SYNC,DATA    "CLEAR BITS 8-15
 OR DATA,TEMP;DPX<DB;DB=SPFN "COMBINE BYTE SWAPPED WORD AND SAVE
 ADD N26,DADR;SETMA;MI<DB;DB=DPX "REPLACE IN MEMORY
 DEC N
 BLT DEMUL
 JMP LOOP    "LOOP UNTIL ALL WORDS SWAPPED
DEMUL: LDSPI BIAS;DB=15.;FADD ZERO,ZERO "SET UP CONSTANTS
 LDSPI TWO ;DB=2. ;FADD
 LDSPI N26 ;DB=26.
 LDSPI SYNC;DB=-1.
 LDSPI MASK;DB=200
 MOV GAIN,GAIN;SETMA   "SET UP SAVE ON DPY
 SUB DINC,DBASE;LDDPA;DB=1 "SET UP BASE ADDRESSES
 LDSPI N;DB=30.
 MOV DBASE,DADR;INCMA
PUSH: DEC N;INCDPA;DPY<MD   "MD-DPY SAVE LOOP
 INCMA;BGT PUSH
 MOV GAIN,GAIN;DPY<SPFN   "SAVE GAIN ADDRESS
 CLR IADR;SETMA    "SET UP MEMORY ACCESS
OUTLP: MOV MASK,TEMP    "PUT MASK IN TEMP
 INC IADR;SETMA
 LDSPI DATA;DB=MD
 SUB SYNC,DATA    "CHECK SYNC BITS
```

```
 LDSPI GAIN;DB=MD;BNE ERR1 "GOTO ERROR IF NE
 AND GAIN,TEMP
 LDSPI DIRCK;DB=1.;BNE SET "CHECK DIRECTION BIT
 MOV SYNC,DIRCK
SET: LDSPI TEMP;DB=31.
 AND TEMP,GAIN   "CLEAR UNWANTED SYNC BITS
 SUB GAIN,TEMP
 ADD BIAS,GAIN;BEQ NEW  "ADD BIAS TO GAIN
 INC GCNT;SETDPA
 LDSPI TEMP;DB=DPY  "GET GAIN TO CHECK
 SUB GAIN,TEMP   "CHECK GAIN AND SUBMUX GAIN
 SUB DIR,DIRCK;BNE ERR2  "CHECK DIRECTION BIT
 LDDPA;DB=1.;BNE ERR3  "SET UP FOR NEXT LOOP
 BR SKIP
ERR1: LDSPI ERR;DB=1.   "SET UP ERROR
 RETURN
ERR2: MOV GAIN,GAIN;DPY<SPFN
 JMP SKIP
ERR3: COM DIR
 INC DIR
 JMP SKIP
NEW: CLR GCNT;LDDPA;DB=1  "HERE EVERY 30 TIMES
SKIP: INC IADR;SETMA
 LDSPI N;DB=30.
CONT: LDSPI GAIN;DB=DPY  "START OF MAIN LOOP
 LDSPI DATA;DB=MD
 MOVR DATA,DATA   "SHIFT DATA WORD R
 BZC SAME
 ADD DIR,GAIN
SAME: MOVL DATA,DATA;DPX<DB;DB=SPFN
 BGE NMI
 ADD TWO,DATA;DPX<DB;DB=SPFN
NMI: MOV N26,N26;FADD ZERO,MDPX
 MOV GAIN,GAIN;DPY<DB;DB=SPFN;FADD
 DPX<FA
 LDSPE TEMP;DB=DPX
 SUB GAIN,TEMP;FADD ZERO,MDPX
 INC IADR;SETMA;FADD
 DEC N;INCDPA
 ADD DINC,DADR;SETMA;MI<FA;BGT CONT
 COM DIR   "TURN -1 TO +1 AND VV
 INC DIR
 INC DBASE
 DEC N1
 MOV DBASE,DADR;BEQ FIN
 JMP OUTLP
FIN: LDDPA;DB=31.
 LDSPI TEMP;DB=DPY
 CLR ERR;LDDPA;DB=1.
 MOV TEMP,TEMP;SETMA;MI<DB;DB=DPY;INCDPA
 LDSPI N;DB=29.
POP: INCMA;MI<DB;DB=DPY;INCDPA;DEC N
 BGT POP
 RETURN
 $END
```

```
 $TITLE VBINSC
 $ENTRY VBINSC,7
" THIS IS A PROGRAM WHICH
" REDUCES THE EXPONENT OF A
" FLOATING POINT NUMBER BY A
" SPECIFIED AMOUNT
"
"    S-PAD DEFINITIONS
 A $EQU 0 "VECTOR BASE ADDRESS
 I $EQU 1 "INC OF VECTOR A
 C $EQU 2 "BASE ADDRESS OF RESULT
 K $EQU 3 "INC OF VECTOR C
 G $EQU 4 "GAIN ADDRESS
 J $EQU 5 "GAIN INCREMENT
 N $EQU 6 "NO OF VECTOR ELEMENTS
 FACT $EQU 7 "GAIN VALUE
 RES $EQU 8 "NEW EXPONENT
VBINSC: MOV G,G;SETMA;FADD ZERO,ZERO "GET GAIN AND INIT FADDER
 MOV A,A;SETMA;FADD  "GET A(0)
 SUB K,C    "SET UP RESULT ADDRESS FOR LOOP
 LDSPI FACT;DB=MD  "GET GAIN ON S-PAD
LOOP: LDSPE RES;DPX<DB;DB=MD  "GET VECT ELEMENT EXPONENT
 ADD J,G;SETMA   "INIT ACCESS TO NEXT G
 SUB FACT,RES;FADD ZERO,MDPX "PUT NEW EXP ON THE NO.
 ADD I,A;SETMA   "INIT ACCESSTO NEXT A
 LDSPI FACT;DB=MD  "GET NEXT GAIN ONTO S-PAD
 DEC N;FADD   "DEC COUNTER AND PUSH FADDER
 ADD K,C;SETMA;MI<FA;BGT LOOP "STORE RES AND GO FOR MORE
 RETURN
 $END
```

To demultiplex SEG-C data tapes recorded in floating point format. This program was written specifically to demultiplex the Seismic Exploration Int. field tapes currently in the possession of the B.G.S. in Edinburgh.

The program is only interactive, i.e. it does not need a parameter file.

To start type:

RUN DK1:SEGCDM

The program will then prompt for input parameters as needed.

NUMBER OF FILES TO DEMULTIPLEX informs the program how many files to process in this run.

TOTAL NUMBER OF TRACES PER FILE give the total number of multiplexed traces, both seismic and auxillary, see SEG-C format.

NUMBER OF SAMPLES PER TRACE trace length in seconds multiplied by sampling frequency.

GIVE READ/WRITE DRIVE NUMBERS the tape drive unit numbers with read and write tapes mounted and on line. Write tape must be write enabled.

GIVE LINE NUMBER this is a number placed in the tape header block for identification

GIVE NUMBER OF GUNS, SOURCE WIDTH, SHOT SPACING thid information is only used in tape header for referance.

GIVE SOURCE/RECEIVER OFFSET AND RECIVER SPACING the correct offset is placed in trace headers for calculation of normal move-out correction.

GIVE NUMBER OF DATA/AUXILLARY TRACES informs the program which traces to indentify as seismic traces.

GIVE AREA NAME used for referance in tape headers only.

GIVE JULLIEN DAY NUMBER again used for referance only.

GIVE YEAR year the data was acquired, referance only.

GIVE OBSERVER NAME name of acqisition ship or company, referance only.

GIVE CURRENT OUTPUT REEL NUMBER for the case where the profile is demultiplexed in stages this number is set to the current output reel number within the line and appears in the tape header for referance.

GIVE NEXT TRACE SEQUENCE NUMBER in SEG-Y every trace has a number starting at 1 for the first trace and incrementing for each succeeding trace. If the profile is demultiplexed in parts then this number has to be set to the next trace sequence number at the start of each session.

The program will then execute the number of files specified. Any errors and information on the termination of the run will be written into a short log file DK6:SEGC.LOG. Currently this exit routine is not fully developed and only gives very sparse information.

As the Seismic Exploration Int. data had different fixed gains applied to different sets of traces, these values are preset in the program and should be checked or removed if the routine is to be used again.

```
C
C  PROGRAM TO DEMULTIPLEX A SEG-C TAPE AND OUTPUT A DURHAM SEG-Y
C  TAPE.
C
C  WRITTEN R.W.HOBBS JULY 1984
C
      VIRTUAL SEIS(4152)
      REAL*4 MTNAM(3)
      INTEGER*2 IRPATT(48),IBUFFA(2096),IBUFFB(2096),TPDRR,
     +TPDRW,ITMERR(2),REELNU,SLSPAC,SROFF,RROFF
      LOGICAL*1 AREANO(24),DAYNO(5),YEARNO(6),OBSVNO(18),OUTBUF(240)
C
      COMMON/INPUT/ NFILES,NTRACE,NSAMP,IWORDW,REELNU,FTRCNO
      COMMON/TAPE/  MTNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/REELHD/LINENO,IGUNNO,SLSPAC,IWIDTH,SROFF,RROFF,ITOTAL,
     +NDATA,NAUX,AREANO,YEARNO,OBSVNO,DAYNO
      COMMON/DATBUF/OUTBUF
      COMMON/SYSVAR/IDRCHR,IDRCHP,IDRIVR,IDRIVP,NDRIVS,NEXDRV
      COMMON/RAWVAR/NRAWBL,NRAWWD,KFULBL,KFULWD,NBLEXP,NWDEXP,
     +NSCEXP,NSCFND
      COMMON/TRDVAR/IRPATT,NRBWDS,NRBPAT,NRBYTS,NRBLKS,KRBA,KRBB
      COMMON/WORKSP/IBUFFA,IBUFFB
C
      DATA NRBWDS/2048/,NRBPAT/48/,NRBYTS/4192/,NRBLKS/8/
      DATA DRHAN/3RDR /
      DATA APHAN/3RAP /
      DATA TPHAN/3RTM /
      DATA MTNAM(1)/3RTM4/,MTNAM(2)/3RTM5/,MTNAM(3)/3RTM6/
      DATA IRPATT/'I ','AM',' A',' M','AT','CH','IN','G ','BI','T ',
     +            'PA','TT','ER','N ','AR','RA','Y ','US','ED',' T',
     +            'O ','DE','TE','CT',' R','EA','D ','OV','ER','SP',
     +            'IL','LS',' I','N ','DO','UB','LE',' B','UF','FE',
     +            'RR','ED',' T','AP','E ','RE','AD','S.'/
C
C  SET UP I/O
C
      IF(IFETCH(DRHAN).NE.0) STOP'DR FETCH ERR'
      IF(IFETCH(APHAN).NE.0) STOP'AP FETCH ERR'
      IF(IFETCH(TPHAN).NE.0) STOP'MT FETCH ERR'
      CALL APINIT(0,0,IAP)
C
C  CALL INPUT ROUTINE
C
      CALL SEGCIN
      ISTOP=0
C
C  LOOKUP OUTPUT DRIVE
C
      CALL SEGCHD(IFILNO,ISTOP)
      IF(ISTOP.NE.0) CALL SEGCEX(IFILNO,ISTOP)
C
C  DEMULTIPLEX
C
      CALL SEGCSB(SEIS,IFILNO,ISTOP)
C
C  EXIT
C
      CALL SEGCEX(IFILNO,ISTOP)
      CLOSE(UNIT=2)
      CALL RESET
      CALL EXIT
      END




      SUBROUTINE SEGCEX(IFILNO,ISTOP)
C
      TYPE *,IFILNO,ISTOP
      RETURN
      END
```

```
      SUBROUTINE SEGCSB(SEIS,IFILNO,ISTOP)
C
C  ROUTINE TO READ AND DEMULTIPLEX A SEG-C TAPE
C
C  SCALE FACTORS APPLIED AT TRACE 12 AND TRACE 40 TO CORRECT EARLY
C  GAINS USED IN SCICON RECORDINGS FROM THE HEBRIDES
C
C  WRITTEN R.W.HOBBS AUG-1984
C
      VIRTUAL SEIS(4152),NBUFF(16400)
      REAL*8 DRFNAM
      REAL*4 MTNAM(3),SCALE(2)
      INTEGER*4 JFNUM,JTRAS,JOFF
      INTEGER*2 IRPATT(48),IBUFFA(2096),IBUFFB(2096),IOVRA(48),
     +IOVRB(48),IUNDRA(48),IUNDRB(48),TPDRR,TPDRW,ITMERR(2),
     +REELNU,THOU,HUN,TEN,UNIT,SLSPAC,SROFF,RROFF,IBUFF(4192)
      LOGICAL*1 AREANO(24),DAYNO(5),YEARNO(6),OBSVNO(18),OUTBUF(240)
C
      COMMON/INPUT/ NFILES,NTRACE,NSAMP,IWORDW,REELNU,FTRCNO
      COMMON/TAPE/  MTNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/REELHD/LINENO,IGUNNO,SLSPAC,IWIDTH,SROFF,RROFF,ITOTAL,
     +NDATA,NAUX,AREANO,YEARNO,OBSVNO,DAYNO
      COMMON/DATBUF/OUTBUF
      COMMON/SYSVAR/IDRCHR,IDRCHP,IDRIVR,IDRIVP,NDRIVS,NEXDRV
      COMMON/RAWVAR/NRAWBL,NRAWWD,KFULBL,KFULWD,NBLEXP,NWDEXP,
     +NSCEXP,NSCFND
      COMMON/TRDVAR/IRPATT,NRBWDS,NRBPAT,NRBYTS,NRBLKS,KRBA,KRBB
      COMMON/WORKSP/IBUFFA,IBUFFB
C
      EQUIVALENCE (IBUFFA(1),IUNDRA(1)),(IBUFFB(1),IUNDRB(1)),
     +(IBUFFA(2049),IOVRA(1)),(IBUFFB(2049),IOVRB(1))
      EQUIVALENCE (OUTBUF(9),JFNUM),(OUTBUF(13),JTRAS),
     +(OUTBUF(29),ITYPE),(OUTBUF(37),JOFF),(OUTBUF(109),IDELAY),
     +(OUTBUF(115),NSMP),(OUTBUF(117),ITIME)
      EQUIVALENCE (IBUFF(1),IBUFFA(1))
C
      DATA DRFNAM/12RDR5SEGC  DAT/
C
C  OPEN DR DISK FILE
C
      CALL DRINIT(DRFNAM,0,IDRCHR)
C
C  GET STAST SHOT NUMBER AND EXTRACT THOUSAND COUNT
C
      TYPE 2500
 2500 FORMAT('$GIVE FIRST SHOT NUMBER ? ')
      ACCEPT 1000,ISHOT
      THOU=IFIX(ISHOT/1000)
      IFILNO=ISHOT-1
      TYPE 2560
 2560 FORMAT('$GIVE NUMBER OF ZERO FILES TO ADD ? ')
      ACCEPT 1000,IPAD
C
C  LOAD SCALE FACTOR
C
      SCALE(1)=4.0
      SCALE(2)=2.0
      CALL APPUT(SCALE,24000,2,2)
      CALL APWD
C
C  START FILE LOOP
C
      N32=NSAMP/32
      N128=NSAMP/128
      DO 900 I=1,NFILES
      IFIL=I
      IDEL=0
      IF(IPAD.GT.0) GOTO 250
C
C  READ FIELD TAPE
C
  100  CALL MTFUNC(IDRIVR,8,IERR,IERREG,IEOTR)
```

```
      IF(IERR.NE.1) GOTO 110
      TYPE 2000,IDRIVR
 2000 FORMAT(' READ DRIVE :',I2,' NOT AVAILIABLE')
      GOTO 120
 110  CALL BUFSET
      CALL CONRD(IERR,IERRMT,IERREG,IEOTR)
      IF(IERR.EQ.0) GOTO 115
      IDEL=-1
      CALL ERRLOG(2,IERR,IERRMT,IERREG)
 115  IF(IEOTR.NE.-1) GOTO 200
      TYPE 2100,IDRIVR
 2100 FORMAT(' LEOT ON READ DRIVE :',I2)
 120  TYPE 2200
 2200 FORMAT('$GIVE NEW READ DRIVE NO.:')
      ACCEPT 1000,IDRIVR
 1000 FORMAT(I6)
      TYPE 2560
      ACCEPT 1000,IPAD
      IF(IPAD.GT.0) GOTO 250
      GOTO 100
C
C  SKIP TAPE MARK
C
 200  CALL CONRD(IERR,IERRMT,IERREG,IEOTR)
      IF(IERR.EQ.0) GOTO 210
      IDEL=-1
      CALL ERRLOG(2,IERR,IERRMT,IERREG)
 210  IF(IEOTR.EQ.-1) GOTO 240
      ISTOP=35
      RETURN
C
C  ARRAY BYTE SWAP DATA ON DR DRIVE
C
 240  IBLKR=0
      DO 600 K=1,N32
      IF(IREADW(4112,IBUFF,IBLKR,IDRCHR).NE.4112)
     +STOP'DR READ ERR'
      CALL ARYSWP(IBUFF(1),4112)
C
C  CHECK FIELD TAPE RECORD HEADER
C
      IF(K.GT.1) GOTO 650
      CALL GETBCD(IBUFF(1),UNIT,TEN,HUN,IDUM)
      IFILNO=(100*HUN)+(10*TEN)+UNIT
      IF(IFILNO.EQ.0) THOU=THOU+1
      IFILNO=(1000*THOU)+IFILNO
      CALL GETBCD(IBUFF(6),ITSMP,UNIT,TEN,HUN)
      IBSCN=(100*HUN)+(10*TEN)+UNITS
      CALL GETBCD(IBUFF(9),IDUM,IDUM,UNIT,TEN)
      IRECL=(10*TEN)+UNIT
      ITSTRT=IBUFF(19)/2
      JSAMP=(1024*IRECL)/ITSMP
      IF(JSAMP.LT.NSAMP) STOP'RECORD LENGTH TOO SHORT'
      IF(IBSCN.LT.4*NTRACE+8) STOP'SCAN LENGTH TOO SHORT'
C
C  WRITE OUT BYTE SWAPPED DATA
C
 650  IF(IWRITW(4096,IBUFF,IBLKR,IDRCHR).NE.4096)
     +STOP'DR WRITE ERR'
      IBLKR=IBLKR+16
 600  CONTINUE
      IF(IWRITW(16,IBUFF(4097),IBLKR,IDRCHR).NE.16)
     +STOP'DR WRITE ERR 2'
C
C  READ DATA AND SORT OUT EACH CHANNEL IN TURN
C
      IF(IDEL.NE.0) TYPE 3100,IFIL,IFILNO
      IF(IDEL.NE.0) WRITE(2,3100) IFIL,IFILNO
 3100 FORMAT(' PROCESSED FILE NO.',I5,' FIELD RECORD NO.',I5,
     +' TAPE READ ERROR')
 250  ITRO=0
      ITADR=NTRACE
```

```
      IF(IPAD.GT.0) IFILNO=IFILNO+1
      DO 800 J=1,NTRACE
      CALL VCLR(12288,1,NSAMP)
      CALL APWR
      IF(IDEL.NE.0) GOTO 740
      IF(IPAD.GT.0) GOTO 740
      ITMER=0
      ITADR=ITADR-1
      IBLKR=0
      IOUT=12288
      DO 700 K=1,N128
      IF(IRDRXM(NBUFF(1),IDRCHR,16400,IBLKR,0).NE.16400)
     +STOP'DR READ ERR 2'
      IBLKR=IBLKR+64
C
C  CHECK TIME CODE
C
      ITCODE=NBUFF(19)/2
      IF(NBUFF(17).EQ."177777.AND.
     + ITSTRT+((K-1)*512).EQ.ITCODE) GOTO 720
      ITMER=ITMER+1
      IF(ITMER.LT.2) GOTO 730
      CALL VCLR(12288,1,NSAMP)
      CALL APWR
      IDEL=1
      WRITE(2,3000) IFIL,IFILNO,J
      TYPE 3000,IFIL,IFILNO,J
 3000 FORMAT(' PROCESSED FILE NO.',I5,' FIELD RECORD NO.',I5,
     +' TRACE NO.',I4,' LOST')
      GOTO 740
C
C  LOAD DATA SCAN INTO AP
C
 720  ITMER=0
 730  CALL APPTXM(NBUFF(17),0,8192,3)
      CALL APWD
      CALL VMOV(2+ITADR,64,IOUT,1,128)
      IOUT=IOUT+128
 700  CALL APWR
C
C  TO REMOVE SCALING FACTOR |||| IF USED AGAIN CHECK
C
      IF(J.GT.12) CALL VSDIV(12288,1,24000,12288,1,NSAMP)
      IF(J.GT.40) CALL VSDIV(12288,1,24001,12288,1,NSAMP)
      GOTO 740
 740  CALL APWR
      CALL APGTXM(SEIS(61),12288,NSAMP,2)
      CALL APWD
C
C  SET UP TRACE HEADER
C
      ITRO=ITRO+1
      CALL JICVT(IFILNO,JFNUM)
      CALL JICVT(ITRO,JTRAS)
      ITYPE=1
      IF(IDEL.NE.0) ITYPE=2
      IF(IPAD.GT.0) ITYPE=3
      XOFF=SROFF+(ITRO-1)*RROFF
      CALL JAFIX(XOFF,JOFF)
      CALL JJCVT(JFNUM)
      CALL JJCVT(JTRAS)
      CALL JJCVT(JOFF)
      NSMP=NSAMP
      ITIME=1000*ITSMP
C
C  WRITE OUT TRACE
C
      CALL TRWRIT(SEIS,IFILNO,IWORDW,FTRCNO,0,ITRO,ISTOP)
      IF(ISTOP.NE.0) RETURN
      IF(IEOTW.EQ.0) GOTO 800
      IEOTW=0
      REELNU=REELNU+1
```

```
      CALL SEGCHD(IFILNO,ISTOP)
      IF(ISTOP.NE.0) RETURN
      GOTO 250
C
C END OT TRACE LOOP
C
 800  CONTINUE
C
C  END OF FILE LOOP
C
      IF(MOD(IFIL,10).EQ.0) TYPE 9000,IFIL
 9000 FORMAT('+NO. OF FILES PROCESSED:',I5)
      IPAD=IPAD-1
 900  CONTINUE
C
C  RETURN
C
      RETURN
      END




      SUBROUTINE SEGCIN
C
C  WRITTEN R.W.HOBBS JULY 1984
C
      REAL*4 MTNAM(3)
      INTEGER*2 IRPATT(48),IBUFFA(2096),IBUFFB(2096),TPDRR,
     +TPDRW,ITMERR(2),REELNU,SLSPAC,SROFF,RROFF
      LOGICAL*1 AREANO(24),DAYNO(5),YEARNO(6),OBSVNO(18)
C
      COMMON/INPUT/ NFILES,NTRACE,NSAMP,IWORDW,REELNU,FTRCNO
      COMMON/TAPE/  MTNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/REELHD/LINENO,IGUNNO,SLSPAC,IWIDTH,SROFF,RROFF,ITOTAL,
     +NDATA,NAUX,AREANO,YEARNO,OBSVNO,DAYNO
      COMMON/SYSVAR/IDRCHR,IDRCHP,IDRIVR,IDRIVP,NDRIVS,NEXDRV
      COMMON/RAWVAR/NRAWBL,NRAWWD,KFULBL,KFULWD,NBLEXP,NWDEXP,
     +NSCEXP,NSCFND
      COMMON/TRDVAR/IRPATT,NRBWDS,NRBPAT,NRBYTS,NRBLKS,KRBA,KRBB
      COMMON/WORKSP/IBUFFA,IBUFFB
C
C  OPEN AND READ INPUT DATA FILE
C
      OPEN(UNIT=1,NAME='DK6:SEGC.DAT',TYPE='OLD',ERR=100)
      READ(1,1000) NFILES,NTRACE,NSAMP
      READ(1,1000) TPDRR,TPDRW
      READ(1,1000) LINENO,IGUNNO,SLSPAC,IWIDTH,SROFF,RROFF,ITOTAL,
     +NDATA,NAUX
      READ(1,1100) AREANO,DAYNO,YEARNO,OBSVNO
 1000 FORMAT(12I6)
 1100 FORMAT(24A1/5A1/6A1/18A1)
      CLOSE(UNIT=1)
      GOTO 200
C
C  INTERACTIVE INPUT
C
 100  TYPE 1500
 1500 FORMAT('$NUMBER OF FILES TO DEMULTIPLEX ? ')
      ACCEPT 1000,NFILES
      TYPE 1510
 1510 FORMAT('$TOTAL NUMBER OF TRACES PER FILE ? ')
      ACCEPT 1000,NTRACE
      TYPE 1520
 1520 FORMAT('$NUMBER OF SAMPLES PER TRACE ? ')
      ACCEPT 1000,NSAMP
      TYPE 1600
 1600 FORMAT('$GIVE READ/WRITE TAPE DRIVE NUMBERS ?,? ')
      ACCEPT 1000,TPDRR,TPDRW
      TYPE 1700
 1700 FORMAT('$GIVE LINE NUMBER ? ')
```

```
      ACCEPT 1000,LINENO
      TYPE 1710
 1710 FORMAT('$GIVE NUMBER OF GUNS,SOURCE WIDTH (M),SHOT SPACING',
     +' (M) ?,?,? ')
      ACCEPT 1000,IGUNNO,IWIDTH,SLSPAC
      TYPE 1720
 1720 FORMAT(' GIVE SOURCE/REC. OFFSET (M),REC. SPACING (M),',
     +'TOTAL REC. LENGTH (M) ?,?,? ')
      ACCEPT 1000,SROFF,RROFF,ITOTAL
      TYPE 1730
 1730 FORMAT('$GIVE NUMBER OF DATA/AUXILARY TRACES ?,? ')
      ACCEPT 1000,NDATA,NAUX
      TYPE 1800
 1800 FORMAT('$GIVE AREA NAME (<24 CHR) ')
      ACCEPT 1200,AREANO
 1200 FORMAT(24A1)
      TYPE 1850
 1850 FORMAT('$GIVE JULLIEN DAY NUMBER ')
      ACCEPT 1250,DAYNO
 1250 FORMAT(5A1)
      TYPE 1900
 1900 FORMAT('$GIVE YEAR ')
      ACCEPT 1300,YEARNO
 1300 FORMAT(6A1)
      TYPE 1950
 1950 FORMAT('$GIVE OBSERVER NAME (<18 CHR) ')
      ACCEPT 1350,OBSVNO
 1350 FORMAT(18A1)
C
C  SET UP CONSTANTS
C
  200  IWORDW=2*NSAMP+120
      IDRIVR=TPDRR
      TYPE 2100
 2100 FORMAT('$GIVE CURRENT OUTPUT REEL NO.? ')
      ACCEPT 1000,REELNU
      TYPE 2200
 2200 FORMAT('$GIVE NEXT TRACE SEQUENCE NO.? ')
      ACCEPT 2250,FTRCNO
 2250 FORMAT(F10.0)
C
C  RETURN
C
      OPEN(UNIT=2,NAME='DK6:SEGC.LOG',TYPE='NEW',ERR=3000)
      RETURN
 3000 STOP'DK6:SEGC.LOG OPEN ERROR'
      END




      SUBROUTINE SEGCHD(IFNUM,IEOT)
C
C TPDRW WRITE DRIVE NUMBER
C
C IFNUM CURRENT FILE NUMBER
C
C IEOT END OF TAPE FLAG
C
      REAL*4 MTNAM(3)
      LOGICAL*1 RLHDBL(80,40),CLIENT(24),LINE(12),AREA(23),
     %REELNO(11),DAY(4),YEAR(5),OBSVER(17),TRINRC(8),AUXRC(8),
     %CDPFLD(10),SMPINT(9),SPINTR(7),BITS(6),BYTES(7),RECFMT(8),
     %RLFMT(8),UNITS(10),SFLTPT(5),GFLTPT(5),SOURCE(12),SORCNO(8),
     %SCPTIN(18),SWIDTH(18),SRCPTN(27),SPDOFS(8),SPDDIS(8),GRPINT(20),
     %GEOGPN(5),PROC1(64),PROC2(64),PROC3(76),PROC4(76),PROC5(76),
     %AREANO(24),DAYNO(5),YEARNO(6),OBSVNO(18),RLHDCD(400)
      INTEGER*2 LINENO,REELNU,NCHAN,NSAMP,IGUNNO,SLSPAC,IWIDTH,
     %SROFF,ITOTAL,TRCNUM,AUXTRS,SAMPIN,ORSINT,NOSMPL,ORSAMP,
     %FMTCD,CDPNO,SORTCD,VERTSM,BINGAN,AMPREC,UNITNO,POLARY,
     %ITMERR(2),RROFF,TPDRR,TPDRW
```

```
      INTEGER*4 LNNOL,RLNOL
C
      COMMON/REELHD/LINENO,IGUNNO,SLSPAC,IWIDTH,SROFF,RROFF,ITOTAL,
     %NDATA,NAUX,AREANO,YEARNO,OBSVNO,DAYNO
      COMMON/TAPE/ MTNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/INPUT/ NFILES,NTRACE,NSAMP,IWORDW,REELNU,FTRCNO
C
      EQUIVALENCE (RLHDBL(11,1),CLIENT(1)),(RLHDBL(9,2),LINE(1)),
     %(RLHDBL(26,2),AREA(1)),(RLHDBL(12,3),REELNO(1)),
     %(RLHDBL(41,3),DAY(1)),(RLHDBL(50,3),YEAR(1)),
     %(RLHDBL(64,3),OBSVER(1)),(RLHDBL(23,5),TRINRC(1)),
     %(RLHDBL(54,5),AUXRC(1)),(RLHDBL(71,5),CDPFLD(1)),
     %(RLHDBL(20,6),SMPINT(1)),
     %(RLHDBL(42,6),SPINTR(1)),(RLHDBL(56,6),BITS(1)),
     %(RLHDBL(74,6),BYTES(1)),(RLHDBL(21,7),RECFMT(1)),
     %(RLHDBL(45,7),RLFMT(1)),(RLHDBL(71,7),UNITS(1)),
     %(RLHDBL(29,8),SFLTPT(1)),(RLHDBL(52,9),GFLTPT(1)),
     %(RLHDBL(17,11),SOURCE(1)),(RLHDBL(41,11),SORCNO(1)),
     %(RLHDBL(63,11),SCPTIN(1)),(RLHDBL(63,12),SWIDTH(1)),
     $(RLHDBL(17,12),SRCPTN(1)),(RLHDBL(19,15),SPDOFS(1)),
     $(RLHDBL(39,15),SPDDIS(1)),(RLHDBL(61,15),GRPINT(1)),
     $(RLHDBL(25,16),GEOGPN(1)),(RLHDBL(17,21),PROC1(1)),
     $(RLHDBL(17,22),PROC2(1)),(RLHDBL(5,23),PROC3(1)),
     $(RLHDBL(5,24),PROC4(1)),(RLHDBL(5,25),PROC5(1))
      EQUIVALENCE (RLHDCD(5),LNNOL),(RLHDCD(9),RLNOL),
     %(RLHDCD(13),TRCNUM),(RLHDCD(15),AUXTRS),(RLHDCD(17),SAMPIN),
     %(RLHDCD(19),ORSINT),(RLHDCD(21),NOSMPL),(RLHDCD(23),ORSAMP),
     %(RLHDCD(25),FMTCD),(RLHDCD(27),CDPNO),(RLHDCD(29),SORTCD),
     %(RLHDCD(31),VERTSM),(RLHDCD(51),BINGAN),(RLHDCD(53),AMPREC),
     %(RLHDCD(55),UNITNO),(RLHDCD(57),POLARY)
C
      DATA RLHDBL/'C',' ','1',' ','C','L','I','E','N','T',24*
     %' ','C','O','M','P','A','N','Y',23*' ',
     %'C','R','E','W',' ','N','O',
     %9*' ','C',' ','2',' ','L','I','N','E',12*' ','A',
     %'R','E','A',24*' ','M','A','P',' ','I','D',26*' ',
     %'C',' ','3',' ','R','E','E','L',' ','N','O',11*' ',
     %'D','A','Y','-',
     %'S','T','A','R','T',' ','O','F',' ','R','E','E','L',5*' ',
     %'Y','E','A','R',
     %6*' ','O','B','S','E','R','V','E','R',18*' ',
     %'C',' ','4',' ','I','N','S','T','R','U','M','E','N','T',':',
     %' ','M','F','G',
     %12*' ','M','O','D','E','L',12*' ','S','E','R','I','A','L',
     %' ','N','O',23*' ',
     %'C',' ','5',' ','D','A','T','A',' ','T','R','A','C','E','S','/',
     %'R','E','C','O','R','D',8*' ',
     %'A','U','X','I','L','I','A','R','Y',' ','T','R','A','C','E',
     %'S','/','R','E','C','O','R','D',9*' ',
     %'C','D','P',' ','F','O','L','D',10*' ',
     %'C',' ','6',' ','S','A','M','P','L','E',' ',
     %'I','N','T','E','R','V','A','L',9*' ',
     %'S','A','M','P','L','E','S','/','T','R','A','C','E',7*' ',
     %'B','I','T','S','/','I','N',6*' ',
     %'B','Y','T','E','S','/','S','A','M','P','L','E',7*' ',
     %'C',' ','7',' ','R','E','C','O','R','D','I','N','G',' ',
     %'F','O','R','M','A','T',8*' ',
     %'F','O','R','M','A','T',' ','T','H','I','S',' ',
     %'R','E','E','L',8*' ',
     %'M','E','A','S','U','R','E','M','E','N','T',' ',
     %'S','Y','S','T','E','M',10*' ',
     %'C',' ','8',' ','S','A','M','P','L','E',' ','C','O','D','E',':',
     %' ','F','L','O','A','T','I','N','G',' ','P','T',5*' ',
     %'F','I','X','E','D',' ','P','T',5*' ',
     %'F','I','X','E','D',' ','P','T','-','G','A','I','N',5*' ',
     %'C','O','R','R','E','L','A','T','E','D',6*' ',
     %'C',' ','9',' ','G','A','I','N',' ',' ','T','Y','P','E',
     %':',' ','F','I','X','E','D',5*' ',
     %'B','I','N','A','R','Y',5*' ',
     %'F','L','O','A','T','I','N','G',' ','P','O','I','N','T',5*' ',
     %'O','T','H','E','R',19*' ',
     %'C','1','0',' ','F','I','L','T','E','R','S',':',' ',
```

```
%'A','L','I','A','S',5* ',
%'H','Z',' ',' ','N','O','T','C','H',5* ','H','Z',
%' ',' ','B','A','N','D',5* ','-',5* ','H','Z',
%' ',' ','S','L','O','P','E',4* ','-',4* ','D','B','/',
%'O','C','T',
%'C','1','1',' ','S','O','U','R','C','E',':',' ',
%'T','Y','P','E',12* ' ,
%'N','U','M','B','E','R','/','P','O','I','N','T',8* ' ,
%'P','O','I','N','T',' ','I','N','T','E','R','V','A','L',18* ' ,
%'C','1','2',5* ' ,'P','A','T','T','E','R','N',':',27* ' ,
%'L','E','N','G','T','H',8* ' ,
%'W','I','D','T','H',18* ' ,
%'C','1','3',' ','S','W','E','E','P',':',' ','S','T','A',
%'R','T',5* ' ,'H','Z',
%' ',' ','E','N','D',5* ','H','Z',
%' ',' ','L','E','N','G','T','H',6* ','M','S',
%' ',' ','C','H','A','N','N','E','L',' ','N','O',5* ' ,
%'T','Y','P','E',8* ' ,
%'C','1','4',' ','T','A','P','E','R',':',' ','S','T','A','R','T',
%' ','L','E','N','G','T','H',7* ' ,
%'M','S',' ',' ','E','N','D',' ','L','E','N','G','T','H',7* ' ,
%'M','S',' ',' ','T','Y','P','E',21* ' ,
%'C','1','5',' ','S','P','R','E','A','D',':',' ','O','F','F',
%'S','E','T',8* ' ,
%'M','A','X',' ','D','I','S','T','A','N','C','E',8* ' ,
%'G','R','O','U','P',' ','I','N','T','E','R','V','A','L',20* ' ,
%'C','1','6',' ','G','E','O','P','H','O','N','E','S',
%':',' ','P','E','R',' ','G','R','O','U','P',5* ' ,
%'S','P','A','C','I','N','G',5* ' ,
%'F','R','E','Q','U','E','N','C','Y',5* ' ,
%'M','F','G',10* ' ,'M','O','D','E','L',7* ' ,
%'C','1','7',5* ' ,'P','A','T','T','E','R','N',':',
%27* ' ,'L','E','N','G','T','H',8* ' ,
%'W','I','D','T','H',18* ' ,
%'C','1','8',' ','T','R','A','C','E','S',' ',
%'S','O','R','T','E','D',' ','B','Y',':',' ',
%'R','E','C','O','R','D',5* ' ,
%'C','D','P',5* ' ,
%'O','T','H','E','R',34* ' ,
%'C','1','9',' ','A','M','P','L','I','T','U','D','E',' ',
%'R','E','C','O','V','E','R','Y',':',' ',
%'N','O','N','E',6* ' ,
%'S','P','H','E','R','I','C','A','L',' ','D','I','V',7* ' ,
%'A','G','C',4* ' ,
%'O','T','H','E','R',14* ' ,
%'C','2','0',' ','M','A','P',' ',
%'P','R','O','J','E','C','T','I','O','N',22* ' ,
%'Z','O','N','E',' ','I','D',7* ' ,
%'C','O','O','R','D','I','N','A','T','E',' ',
%'U','N','I','T','S',10* ' ,
%'C','2','1',' ','P','R','O','C','E','S','S','I','N','G',
%':',65* ' ,
%'C','2','2',' ','P','R','O','C','E','S','S','I','N','G',
%':',65* ' ,
%'C','2','3',77* ' ,'C','2','4',77* ' ,'C','2','5',77* ' ,
%'C','2','6',77* ' ,'C','2','7',77* ' ,'C','2','8',77* ' ,
%'C','2','9',77* ' ,'C','3','0',77* ' ,'C','3','1',77* ' ,
%'C','3','2',77* ' ,'C','3','3',77* ' ,'C','3','4',77* ' ,
%'C','3','5',77* ' ,'C','3','6',77* ' ,'C','3','7',77* ' ,
%'C','3','8',77* ' ,'C','3','9',77* ' ,
%'C','4','0',' ','E','N','D',' ',
%'A','S','C','I','I',67* ' /
 DATA PROC1/'D','E','M','U','L','T','I','P','L','E','X','E','D',
%51* ' /
 DATA CLIENT/' ','D','U','R','H','A','M',' ','U','N',
%'I','V','E','R','S','I','T','Y',6* ' /
 DATA CDPFLD/' ','1',8* ' /
 DATA SMPINT/' ','4','0','0','0',4* ' /
 DATA BITS/' ','1','6','0','0',' ' /
 DATA BYTES/' ','4',5* ' /
 DATA RECFMT/' ','P','E',5* ' /
 DATA RLFMT/' ','P','E',5* ' /
```

```
      DATA UNITS/' ','M','E','T','R','E','S',3*' '/
      DATA SFLTPT/' ','Y','E','S',' '/
      DATA GFLTPT/' ','Y','E','S',' '/
      DATA SOURCE/' ','A','I','R','-','G','U','N','S',3*' '/
      DATA SRCPTN/' ','L','I','N','E','A','R',20*' '/
C
C     ASSIGN CHANNEL NUMBERS
C
 210  ICHN=TPDRW+10
C
C     LOOKUP TAPE DRIVE
C
      TYPE 2000,TPDRW,IFNUM
 2000 FORMAT(' START NEW WRITE DRIVE:',I1,' FILE NUMBER:'I4)
      IEOTW=0
      ILK=LOOKUP(ICHN,MTNAM(TPDRW+1),-1)
      IER=IPEEK("172520)
      ISTUS=IER.AND."100
      IF(ISTUS.GT.0) GOTO 240
      TYPE 2100,TPDRW,IFNUM
 2100 FORMAT(' GIVEN DRIVE NO:'I1,' NOT AVAILABLE AT FILE NO:',I4)
      PAUSE
      GOTO 210
C
C     CHECK WRITE DRIVE IS NOT WRITE PROTECTED
C
 240  ISTUS=IER.AND."4
      IF(ISTUS.EQ.0) GOTO 220
      TYPE 2900,TPDRW,IFNUM
 2900 FORMAT(' GIVEN WRITE DRIVE NO:',I1,' IS WRITE PROTECTED')
      PAUSE
      GO TO 210
C
C WRITE TAPE HEADER BLOCKS IF AT START OF TAPE
C
 220  ISTUS=IER.AND."40
      IF(ISTUS.EQ.0) RETURN
      DO 100 I=1,23
 100  AREA(I)=AREANO(I)
      DO 130 I=1,4
 130  DAY(I)=DAYNO(I)
      DO 110 I=1,5
 110  YEAR(I)=YEARNO(I)
      DO 120 I=1,17
 120  OBSVER(I)=OBSVNO(I)
      ENCODE(12,1010,LINE,ERR=30) LINENO
      ENCODE(11,1010,REELNO,ERR=30) REELNU
      ENCODE(8,1010,TRINRC,ERR=30) NDATA
      ENCODE(8,1010,AUXRC,ERR=30) NAUX
      ENCODE(7,1010,SPINTR,ERR=30) NSAMP
 1010 FORMAT(I5)
      ENCODE(8,1010,SORCNO,ERR=30) IGUNNO
      ENCODE(18,1010,SCPTIN,ERR=30) SLSPAC
      ENCODE(18,1010,SWIDTH,ERR=30) IWIDTH
      ENCODE(8,1010,SPDOFS,ERR=30) SROFF
      ENCODE(20,1010,GRPINT,ERR=30) RROFF
      ENCODE(8,1010,SPDDIS,ERR=30) ITOTAL
      GO TO 40
 30   STOP 'ENCODE ERROR'
 40   IF(IWMTLM(RLHDBL(1,1),ICHN,1600,ITMERR(1)).NE.1600) STOP 'ERROR
     + WRITING ASCII TAPE HEADER'
      CALL ISPFNW("377,ICHN)
      DO 101 M=1,400
 101  RLHDCD(M)=0
      TRCNUM=NDATA
      AUXTRS=NAUX
      SAMPIN=4000
      ORSINT=4000
      FMTCD=1
      CDPNO=1
      SORTCD=1
      VERTSM=1
```

```
      BINGAN=1
      AMPREC=1
      UNITNO=1
      POLARY=3
      NOSMPL=NSAMP
      ORSAMP=NSAMP
      TRCNUM=NCHAN
      CALL JICVT(LINENO,LNNOL)
      CALL JICVT(REELNU,RLNOL)
      CALL JJCVT(LNNOL)
      CALL JJCVT(RLNOL)
      IF(IWMTLM(RLHDCD(1),ICHN,200,ITMERR(1)).NE.200) STOP 'ERROR
     + WRITING BINARY TAPE HEADER'
      CALL ISPFNW("377,ICHN)
      RETURN
      END




C
C          " ROUTINE DRINIT
C
C            A. N. BOWEN MARCH 1984
C
C            CREATES DISK FILE OF SPECIFIED NAME AND SIZE ON DR DRIVE.
C------------
C            CALL FORMAT:
C
C            CALL DRINIT(DRFNAM,NDFBLK,IDRCH)
C            DRFNAM -  REAL*8 FILENAME
C            NDFBLK -  NUMBER OF BLOCKS REQUIRED IN FILE
C            IDRCH  -  RETURNED CHANNEL NUMBER ALLOCATED
C
C            NEEDS DISK HANDLER PREVIOUSLY FETCHED
C------------
      SUBROUTINE DRINIT(DRFNAM,NDFBLK,IDRCH)
      REAL*8 DRFNAM
      IDRCH=IGETC()
      IF(IDRCH.LT.0)STOP 'NO CHANNEL FREE'
      IEN=IENTER(IDRCH,DRFNAM,NDFBLK)
      IF(IEN.GE.0)GO TO 50
      GO TO (10,20,30,40), IEN+5
10    STOP 'DRINIT - ATTEMPT TO OVERWRITE PROTECTED FILE'
20    STOP 'DRINIT - DEVICE IN USE'
30    STOP 'DRINIT - NO ROOM ON DR DISK'
40    STOP 'DRINIT - CHANNEL IN USE'
50    RETURN
      END




C
C          " ROUTINE TIDY
C
C            A.N. BOWEN MAY 1984
C
C            USED WITH DOUBLE-BUFFERED TAPE READ TO DETECT OVERSPILL.
C            LOOKS AT THE DISRUPTION TO A PRESET BIT-PATTERN AT THE TOP
C            OF THE BUFFER ARRAYS IN COMMON AND COPIES EXTRA WORDS
C            WRITTEN PAST THE NOMINAL BUFFER END TO THEIR CORRECT LOCATION
C            AT THE START OF THE OTHER BUFFER. THE UNDISTURBED AREA OF THE
C            BUFFER BEING EXAMINED IS IDENTIFIED BY A TWO-WORD MATCH WITH
C            THE PRESET BIT-PATTERN, WHICH IS RESTORED ON EXIT.
C------------
C            CALL FORMAT:
C
C            CALL TIDY(IAORB,IERR)
C            IAORB  -  IF=0, TESTS FOR SPILLAGE PAST END OF BUFFER A.
```

```
C                       OTHERWISE TESTS BUFFER B
C             IERR   -  IF=0, SUCCESSFUL.
C                       IF=1, NO MATCH FOUND.
C-------------
C
      SUBROUTINE TIDY(IAORB,IERR)
      LOGICAL*1 P,C
C
      COMMON /TRDVAR/ IRPATT(48),NRBWDS,NRBPAT,NRBYTS,NRBLKS,KRBA,KRBB
      COMMON /WORKSP/ IBUFFA(2096),IBUFFB(2096)
C
      DIMENSION IOVRA(48),IOVRB(48),IUNDRA(48),IUNDRB(48)
C
      EQUIVALENCE (IBUFFA(1),IUNDRA(1)),(IBUFFB(1),IUNDRB(1)),
     .            (IBUFFA(2049),IOVRA(1)),(IBUFFB(2049),IOVRB(1))
C
      IERR=0
      P=.FALSE.
      IF(IAORB.EQ.0)GO TO 100
      DO 10 IWD=1,NRBPAT
      C=.FALSE.
      IF(IOVRB(IWD).EQ.IRPATT(IWD))C=.TRUE.
      IF(P.AND.C)GO TO 20
      P=C
10    CONTINUE
      IERR=1
      RETURN
20    NREPS=IWD-2
      IF(NREPS.EQ.0)RETURN
      DO 30 IWD=1,NREPS
      IUNDRA(IWD)=IOVRB(IWD)
      IOVRB(IWD)=IRPATT(IWD)
30    CONTINUE
      RETURN
100   CONTINUE
      DO 110 IWD=1,NRBPAT
      C=.FALSE.
      IF(IOVRA(IWD).EQ.IRPATT(IWD))C=.TRUE.
      IF(P.AND.C)GO TO 120
      P=C
110   CONTINUE
      IERR=1
      RETURN
120   NREPS=IWD-2
      IF(NREPS.EQ.0)RETURN
      DO 130 IWD=1,NREPS
      IUNDRB(IWD)=IOVRA(IWD)
      IOVRA(IWD)=IRPATT(IWD)
130   CONTINUE
      RETURN
      END




C             ROUTINE ERRLOG
C
C             A.N. BOWEN JUN 1984
C
C             LOGS ERRORS FROM SEG-B DEMUX ROUTINE
C-------------
C             CALL FORMAT:
C
C             CALL ERRLOG(ITYPE,IERR,IERRMT,IERREG)
C             ARGUMENTS ARE ERROR CODES RETURNED THROUGH
C             LEVELS OF SUBROUTINE CALLS.
C-------------
C
      SUBROUTINE ERRLOG(ITYPE,IERR,IERRMT,IERREG)
C
```

```
      TYPE 9000, ITYPE,IERR,IERRMT,IERREG
9000  FORMAT(//,' ******      ERROR LOG CODES',4010,/)
      GO TO (100,200,300,400), ITYPE
C
100   TYPE *,'******      HEADRD ERROR'
      GO TO (210,220,110,120,130), IERR
110   TYPE *,'******      Filemark read'
      GO TO 10000
120   TYPE *,'******      Incorrect header length'
      TYPE *,'******      Number of empty words in buffer=',IERREG
      GO TO 10000
130   TYPE *,'******      Header longer than buffer (4096 bytes)'
      GO TO 10000
C
200   TYPE *,'******      CONRD  ERROR'
      GO TO (210,220,230,240), IERR
210   TYPE *,'******      Traced to GOTAPL'
      TYPE *,'******      Tape drive error (not EOF or EOT)'
      TYPE 9100, IERRMT
9100  FORMAT(' ******      MTS register octal',O12)
      GO TO 10000
220   TYPE *,'******      Traced to SWITCH'
      GO TO (221,222,223), IERRMT
221   TYPE *,'******      Read completed (not considered an error)'
      TYPE *,'******      Number of empty words in buffer=',IERREG
      GO TO 10000
222   TYPE *,'******      MTBC overflow (did not return in time)'
      GO TO 10000
223   TYPE *,'******      Tape drive error (not EOF or EOT)'
      TYPE 9100, IERREG
      GO TO 10000
230   TYPE *,'******      Traced to disk write error'
      GO TO 10000
240   TYPE *,'******      Traced to TIDY'
      GO TO 10000
C
300   TYPE *,'******      CONDMX ERROR'
      GO TO (310,320,330,340,230), IERR
310   TYPE *,'******      Traced to BREAK'
      GO TO (311,312), IERRMT
311   TYPE *,'******      No non-zero words in file'
      GO TO 10000
312   TYPE *,'******      Disk read error'
      GO TO 10000
320   TYPE *,'******      Traced to SIZEF'
      GO TO (321,322), IERRMT
321   TYPE *,'******      Not an integral number of scans in file'
      GO TO 10000
322   TYPE *,'******      Incorrect number of scans in file'
      GO TO 10000
330   TYPE *,'******      Traced to disk read error'
      GO TO 10000
340   TYPE *,'******      Traced to DMXB'
      GO TO 10000
C
400   TYPE *,'******      CONWRT ERROR'
      GO TO (410,420), IERRMT
410   TYPE *,'******      Traced to DMXRHD'
      GO TO 10000
420   TYPE *,'******      Traced to TRANS'
10000 IERR=0
      IERRMT=0
      IERREG=0
      RETURN
      END
```

```
C
C          " ROUTINE BUFSET
C
C            A.N. BOWEN MAY 1984
C
C            GETS ADDRESSES OF COMMON ARRAYS USED FOR DOUBLE-
C            BUFFERED TAPE READ AND CALCULATES CONSTANTS REQUIRED FOR
C            BUFFER SWITCHING. ALSO PRESETS BIT PATTERN PAST THE END OF
C            EACH ARRAY, USED TO DETECT READ OVERSPILLS.
C            N.B. IN ORDER TO PREVENT THE XM BITS OF THE TAPE REGISTERS
C            FROM INCREMENTING IN RESPONSE TO CARRY-BITS FROM OPERATIONS
C            PERFORMED ON MTCMA IT IS ESSENTIAL THAT THE READ BUFFERS IN
C            COMMON DO NOT STRADDLE THE 32-KBYTE ADDRESS BOUNDARY.
C            BUFSET CHECKS THIS, AND ABORTS IF NECESSARY.
C-------------
C            CALL FORMAT:
C
C            CALL BUFSET
C-------------
C
      SUBROUTINE BUFSET
C
      COMMON /TRDVAR/ IRPATT(48),NRBWDS,NRBPAT,NRBYTS,NRBLKS,KRBA,KRBB
      COMMON /WORKSP/ IBUFFA(2096),IBUFFB(2096)
C
      DIMENSION IOVRA(48),IOVRB(48),IUNDRA(48),IUNDRB(48)
C
      EQUIVALENCE (IBUFFA(1),IUNDRA(1)),(IBUFFB(1),IUNDRB(1)),
     .            (IBUFFA(2049),IOVRA(1)),(IBUFFB(2049),IOVRB(1))
C
      IBAST=IADDR(IBUFFA(1))
      IBAND=IADDR(IBUFFA(NRBWDS))
      IBBST=IADDR(IBUFFB(1))
      IBBND=IADDR(IBUFFB(NRBWDS))
      IF(IBAST.LT.0.AND.IBAND.LT.0.AND.IBBST.LT.0.AND.IBBND.LT.0)
     .GO TO 100
      IF(IBAST.GE.0.AND.IBAND.GE.0.AND.IBBST.GE.0.AND.IBBND.GE.0)
     .GO TO 100
      TYPE 9000
      TYPE 9100, IBAST,IBAND,IBBST,IBBND
      STOP
100   KRBA=(IBAND-IBBST)+2
      KRBB=(IBBND-IBAST)+2
      DO 10 IWD=1,NRBPAT
      IOVRA(IWD)=IRPATT(IWD)
      IOVRB(IWD)=IRPATT(IWD)
10    CONTINUE
      RETURN
9000  FORMAT(' ******  READ BUFFERS STRADDLE 32 KBYTE BOUNDARY  ******')
9100  FORMAT(' ******  ADDRESSES ARE:',4O10)
      END
```

```
C
C          " ROUTINE CONRD
C
C            A.N. BOWEN MAY 1984
C
C            ROUTINE CONTROLS DOUBLE-BUFFERED TAPE READ. USES
C            SIMULTANEOUS TAPE READ/DR DISK WRITE OPERATIONS TO
C            ALLOW A TAPE BLOCK OF ARBITRARY SIZE TO BE READ AND
C            TRANSFERRED TO DISK. REQUIRES TWO LOW-MEMORY BUFFERS
C            OF 2096 WORDS EACH, WHICH ARE HELD IN COMMON.
C-------------
C            CALL FORMAT:
C
C            CALL CONRD(IERR,IERRMT,IERREG,IEOTR)
C            IERR   -  IF=0, NO ERROR
C                      IF=1, GOTAPL ERROR
C         IF=2, SWITCH ERROR
```

```
C          IF=3, DISK WRITE ERROR
C          IF=4, TIDY ERROR
C       IERRMT -  ERROR CODE PASSED FROM SUBROUTINE
C       IERREG -  MTS REGISTER CONTENTS PASSED FROM SUBROUTINE
C       IEOTR  -  SET TO +1 IF EOT ENCOUNTERED DURING READ
C          SET TO -1 IF FILEMARK ENCOUNTERED DURING READ
C          SET TO  0 OTHERWISE
C-------------
C
      SUBROUTINE CONRD(IERR,IERRMT,IERREG,IEOTR)
C
      COMMON /SYSVAR/ IDRCHR,IDRCHP,IDRIVR,IDRIVP,NDRIVS,NEXDRV
      COMMON /RAWVAR/ NRAWBL,NRAWWD,KFULBL,KFULWD,NBLEXP,NWDEXP,
     .                NSCEXP,NSCFND
      COMMON /TRDVAR/ IRPATT(48),NRBWDS,NRBPAT,NRBYTS,NRBLKS,KRBA,KRBB
      COMMON /WORKSP/ IBUFFA(2096),IBUFFB(2096)
C
      DIMENSION IOVRA(48),IOVRB(48),IUNDRA(48),IUNDRB(48)
C
      EQUIVALENCE (IBUFFA(1),IUNDRA(1)),(IBUFFB(1),IUNDRB(1)),
     .            (IBUFFA(2049),IOVRA(1)),(IBUFFB(2049),IOVRB(1))
      NRAWBL=0
      IERREG=0
      CALL GOTAPL(IDRIVR,1,IBUFFA(1),NRBYTS,IERRMT)
      IF(IERRMT.NE.0)GO TO 900
100   CALL SWITCH(IBUFFA(NRBWDS),KRBA,NRBYTS,IERRMT,IERREG,IEOTR)
      IF(IERRMT.EQ.0)GO TO 110
      IF(IERRMT.NE.1)GO TO 910
      NRAWWD=NRBWDS-IERREG
      IF(NRAWWD.LE.0)GO TO 1000
      IF(IWRITW(NRAWWD,IBUFFA,NRAWBL,IDRCHR).LT.0)GO TO 920
      GO TO 1000
110   CALL TIDY(0,IERRMT)
      IF(IERRMT.NE.0)GO TO 930
      IF(IWRITW(NRBWDS,IBUFFA,NRAWBL,IDRCHR).LT.0)GO TO 920
      NRAWBL=NRAWBL+NRBLKS
      CALL SWITCH(IBUFFB(NRBWDS),KRBB,NRBYTS,IERRMT,IERREG,IEOTR)
      IF(IERRMT.EQ.0)GO TO 120
      IF(IERRMT.NE.1)GO TO 910
      NRAWWD=NRBWDS-IERREG
      IF(NRAWWD.LE.0)GO TO 1000
      IF(IWRITW(NRAWWD,IBUFFB,NRAWBL,IDRCHR).LT.0)GO TO 920
      GO TO 1000
120   CALL TIDY(1,IERRMT)
      IF(IERRMT.NE.0)GO TO 930
      IF(IWRITW(NRBWDS,IBUFFB,NRAWBL,IDRCHR).LT.0)GO TO 920
      NRAWBL=NRAWBL+NRBLKS
      GO TO 100
1000  CONTINUE
      IERR=NRAWWD/256
      NRAWBL=NRAWBL+IERR
      NRAWWD=NRAWWD-(IERR*256)
      IERR=0
      IERRMT=0
      IERREG=0
      RETURN
900   IERR=1
      RETURN
910   IERR=2
      RETURN
920   IERR=3
      IERRMT=NRAWBL
      RETURN
930   IERR=4
      RETURN
      END
```

```
;
;  ROUTINE SWITCH
;
; A.N. BOWEN MARCH 1984
;
; USED WITH DOUBLE-BUFFERED TAPE READ PROGRAM TO SWITCH DESTINATION
; OF WORDS READ FROM TAPE BETWEEN BUFFERS IN LOW MEMORY. RESETS
; MTCMA AFTER IT PASSES THE END OF THE CURRENT BUFFER TO POINT AT
; AN EQUIVALENT LOCATION IN THE NEW BUFFER (REGARDLESS OF THE ACTUAL
; VALUE OF MTCMA AT THE TIME OF RESET). OVERSPILL PAST THE CURRENT
; BUFFER END IS MOVED TO ITS CORRECT LOCATION AT THE START OF THE
; NEW BUFFER BY FORTRAN ROUTINE TIDY. IF THE READ OPERATION HAS
; ALREADY TERMINATED ON ENTRY, OR TERMINATES BEFORE THE SWITCH,
; THE NUMBER OF EMPTY WORDS IN THE CURRENT BUFFER IS CALCULATED
; AND RETURNED.
;
; CALL FORMAT:
;
; CALL SWITCH(IBUFND,ICONS,NBYTES,IERR,IERREG,IEOT)
; IBUFND  -  LAST ELEMENT OF CURRENT ARRAY
; ICONS   -  CONSTANT SUBTRACTED FROM MTCMA
;      (OFFSET BETWEEN END OF CURRENT BUFFER AND START OF NEW ONE)
; NBYTES  - NUMBER OF BYTES IN BUFFER + EXTRA BYTES FOR OVERSPILL
;       IERR    -  ERROR CODE:  0=NO ERROR
;     1=READ COMPLETED
;     2=MTBC OVERFLOW (DID NOT RETURN IN TIME)
;     3=ERROR DETECTED (NOT EOF OR EOT)
; IERREG  -  IF NO ERROR, ZERO
;      IF ERROR TYPE 1, NUMBER OF EMPTY WORDS IN BUFFER
;      IF OTHER ERROR, CONTENTS OF MTS ON EXIT
; IEOT    -  SET TO +1 IF EOT ENCOUNTERED DURING READ
;      SET TO -1 IF FILEMARK ENCOUNTERED DURING READ
;      SET TO  0 OTHERWISE
;
;-------
 .TITLE SWITCH
 .GLOBL SWITCH
;
 MTS = 172520 ; ADDRESS OF MTS REGISTER
 MTCMA = 172526 ; ADDRESS OF MTCMA REGISTER
;
SWITCH: TST (R5)+   ; SKIP NUMBER OF ARGUMENTS
 MOV (R5)+,R0 ; GET IBUFND ADDRESS
 MOV @(R5)+,R1 ; GET ICONS
 MOV @(R5)+,R2 ; GET NBYTES
 NEG R2   ; 2'S COMPLEMENT
 MOV #MTS,R3   ; MTS ADDRESS
 MOV #MTCMA,R4 ; MTCMA ADDRESS
NOTYET: BIT #1,(R3)   ; DECK BUSY?
 BNE ERR1   ; NO, ERROR TYPE 1
 CMP (R4),R0   ; MTCMA PASSED IBUFND?
 BLOS NOTYET   ; LOOP IF BUFFER END NOT PASSED YET
RESET: SUB R1,(R4)   ; SUBTRACT ICONS FROM MTCMA
 MOV R2,4(R3) ; RESET MTBC TO -NBYTES
        CLR     @(R5)+          ; SET IERR=0
 CLR @(R5)+   ; CLEAR IERREG
 BR CLEAR
ERR1: TST 4(R3)   ; MTBC OVERFLOW?
 BPL ERR2   ; YES, ERROR TYPE 2
 CMP (R4),R0   ; MTCMA PASSED IBUFND?
 BHI RESET   ; YES, NEEDS TO BE TIDIED UP
 MOV #1,@(R5)+ ; SET IERR=1
 SUB (R4),R0   ; IBUFND-MTCMA
 ASR R0   ; DIVIDED BY 2
 INC R0   ; PLUS 1
 MOV R0,@(R5)+ ; NUMBER OF EMPTY WORDS TO IERREG
 TST 2(R3)   ; ERROR?
 BPL CLEAR   ; NO, EXIT
 BIT #2000,(R3) ; EOT?
 BNE EOT   ; YES, BRANCH
 BIT #40000,(R3) ; EOF?
 BNE EOF   ; YES, BRANCH
```

```
 BIT #10000,(R3) ; PARITY ERROR?
 BNE CLEAR  ; YES, EXIT
 MOV #3,@-4(R5) ; RESET IERR=3
 MOV (R3),@-2(R5) ; PUT MTS TO IERREG
 BR CLEAR
ERR2: MOV #2,@(R5)+ ; IERR=2
 CLR @(R5)+  ; CLEAR IERREG
CLEAR: CLR @(R5)  ; IEOT=0
 RTS PC
EOT: MOV #1,@(R5) ; SET EOT=+1
 BIS #1400,2(R3) ; CLEAR FORMATTER EOT LATCH USING INVALID DRIVE
 RTS PC
EOF: MOV #-1,@(R5) ; SET EOT=-1
 RTS PC
        .END
```

```
;
; ROUTINE GETBCD
;
; A.N. BOWEN JUN 1984
;
; ROUTINE DECODES 4 BCD DIGITS FROM 2-BYTE WORD.
;--------
; CALL FORMAT:
;
; CALL GETBCD(IWORD,I1,I2,I3,I4)
; IWORD  -  WORD CONTAINING 4 BCD DIGITS
; I1     -  DIGIT IN BITS 0 - 3
; I2     -  DIGIT IN BITS 4 - 7
; I3     -  DIGIT IN BITS 8 -11
; I4     -  DIGIT IN BITS 12-15
;--------
 .TITLE GETBCD
 .GLOBL GETBCD
;
GETBCD: TST (R5)+  ; SKIP NUMBER OF ARGUMENTS
 MOV #4,R2  ; LOAD DIGIT COUNTER
 MOV @(R5)+,R0 ; GET WORD TO BE DECODED
LOOP: MOV R0,R1  ; COPY
 BIC #177760,R1 ; MASK OFF BITS 4-15
 MOV R1,@(R5)+ ; PUT DIGIT TO ARGUMENT LIST
 ASH #-4,R0  ; SHIFT WORD RIGHT 4
 DEC R2  ; DECREMENT DIGIT COUNT
 BNE LOOP  ; NEXT DIGIT IF NEEDED
 RTS PC
 .END
```

```
;
;  ROUTINE GOTAPL
;
; A.N. BOWEN MARCH 1984
;
; INITIATES A TAPE READ/WRITE OF GIVEN BYTE LENGTH TO
;       OR FROM A FORTRAN-DECLARED LOW-MEMORY ARRAY
;--------
;       CALL FORMAT:
;
;       CALL GOTAPL(IDRIV,IFUNC,IBUFAD,NBYTES,IERR)
;       IDRIV  -  TAPE DRIVE NUMBER (0,1,2)
;       IFUNC  -  FUNCTION CODE: READ (1) OR WRITE (2)
;       IBUFAD -  FIRST ELEMENT OF ARRAY TO BE READ OR WRITTEN TO
;       NBYTES -  NUMBER OF BYTES TO BE TRANSFERRED
;       IERR   -  RETURNED ERROR FLAG. IF ZERO, NO ERROR. OTHERWISE
;                 CONTAINS IMAGE OF MTS REGISTER
```

```
;
;       RETURNS TO CALLING PROGRAM AS SOON AS READ/WRITE INITIATED
;-------
 .TITLE GOTAPL
 .GLOBL GOTAPL
;
 MTS = 172520 ; ADDRESS OF MTS REGISTER
;
GOTAPL: TST (R5)+   ; SKIP NUMBER OF ARGUMENTS
 CLR R0   ; SET 9-TRACK, 1600BPI, ODD PARITY
;       CLEAR XM BITS & DISABLE I'RRUPTS
 MOV @(R5)+,R1 ; GET IDRIV
 ASH #10,R1   ; SHIFT LEFT 8
        BIS     R1,R0           ; OR WITH BITMASK
 MOV @(R5)+,R1 ; GET IFUNC
 ASL R1   ; SHIFT LEFT 1
        BIS     R1,R0           ; OR WITH BITMASK
 MOV #MTS,R3   ; LOAD BASE ADDRESS
 MOV R0,2(R3) ; LOAD BITMASK INTO MTC
WAIT: BIT #1,(R3)   ; READY?
 BEQ WAIT   ; NO, LOOP
 MOV (R5)+,6(R3) ; LOAD MTCMA WITH ARRAY START ADDRESS
 MOV @(R5)+,R2 ; GET NBYTES
 NEG R2   ; 2'S COMPLEMENT
 MOV R2,4(R3) ; LOAD MTBC WITH -NBYTES
 BIS #1,R0   ; SET GO-BIT IN BITMASK
 MOV R0,2(R3) ; INITIATE TAPE ACTION
 TST 2(R3)   ; ERROR?
 BMI ERR   ; YES, BRANCH
OUT: CLR @(R5)   ; IERR=0
 RTS PC
ERR: BIT #42000,(R3) ; COMPLAINING ABOUT FILEMARK OR EOT?
 BNE OUT   ; YES, IGNORE IT
 MOV (R3),@(R5) ; GET MTS TO IERR
 RTS PC   ; RETURN TO FORTRAN
        .END



; ROUTINE MTFUNC
;
; A.N. BOWEN MARCH 1984
;
; MINIMAL VERSION OF SKELETON MT HANDLER MTUTIL.MAC
; PERFORMS BASIC MT FUNCTIONS BY ACCESSING DEVICE REGISTERS DIRECTLY
;-------
;
; CALL FORMAT:
; CALL MTFUNC(IDRIV,IFUNC,IERR,IERREG,IEOT)
; IDRIV  -  TAPE DRIVE NUMBER
; IFUNC  -  IF=0, WILL REWIND TO BOT BUT REMAIN ONLINE
;     IF=1, WILL SPACE ONE BLOCK FORWARDS
;     IF=2, WILL SPACE ONE BLOCK BACKWARDS
;     IF=7, WILL WRITE A FILEMARK
;     IF=8, WILL CHECK ONLINE, SEL/RDY, AND SET IEOT CORRECTLY
;     IF=9, WILL REWIND TO BOT AND SWITCH OFFLINE
;     (N.B. THESE CODES ARE ARE SUBSET OF THOSE FOR MTUTIL)
; IERR   -  ERROR CODE:
;    0 = NO ERROR
;    1 = DECK NOT ONLINE
;    2 = UNDEFINED FUNCTION
;    3 = ERROR DURING FUNCTION (NOT EOF OR EOT)
; IERREG - CONTENTS OF MTS REGISTER IF ERROR
; IEOT   - SET TO +1 IF EOT DETECTED
;    SET TO -1 IF EOF DETECTED
;    SET TO  0 OTHERWISE
;
 .TITLE MTFUNC
 .GLOBL MTFUNC
;
```

```
 MTS = 172520 ; BASE ADDRESS FOR TAPE REGISTERS
;
MTFUNC: TST (R5)+   ; SKIP NUMBER OF ARGUMENTS
 CLR R0   ; START MAKING BITMASK
 MOV @(R5)+,R1 ; GET IDRIV
 ASH #10,R1   ; SHIFT LEFT 8
 BIS R1,R0   ; OR WITH BITMASK
 MOV #MTS,R3   ; ADDRESS OF MTS REGISTER
CURDY: BIT #200,2(R3) ; CONTROLLER READY?
 BEQ CURDY   ; NO, LOOP
 MOV R0,2(R3) ; LOAD BITMASK INTO MTC
SELRDY: BIT #100,(R3) ; DECK ONLINE?
 BEQ ERR1   ; NO, ERROR TYPE 1
TURDY: BIT #1,(R3)   ; DECK READY?
 BEQ TURDY   ; NO, LOOP
 BIS #10000,2(R3) ; POWER CLEAR PART 1
 BIC #10000,2(R3) ; POWER CLEAR PART 2
 MOV @(R5)+,R1 ; GET IFUNC
 CMP R1,#11   ; AND BRANCH TO FUNCTION
 BEQ REWOFF
 CMP R1,#0
 BEQ REWIND
 CMP R1,#1
 BEQ SKIPBL
 CMP R1,#2
 BEQ BACKBL
 CMP R1,#7
 BEQ WFM
 CMP R1,#10
 BNE ERR2   ; UNIDENTIFIABLE FUNCTION
 MOV R0,2(R3) ; RELOAD DRIVE NUMBER TO MTC
 BIT #2000,(R3) ; EOT?
 BNE EOT   ; YES, BRANCH
 BIT #40000,(R3) ; EOF?
 BNE EOF   ; YES, BRANCH
 CLR @4(R5)   ; CLEAR IEOT
 BR EXIT
REWOFF: BIS #1,R0   ; LOAD CODE
 MOV R0,2(R3) ; LOAD MTC
 BR EXIT
REWIND: BIS #17,R0   ; LOAD CODE
 MOV R0,2(R3) ; LOAD MTC
 BR EXIT
SKIPBL: MOV #11,R1   ; LOAD CODE
 MOV #-1,4(R3) ; LOAD MTBC
 BR GO
BACKBL: MOV #13,R1   ; LOAD CODE
 MOV #-1,4(R3) ; LOAD MTBC
 BR GO
WFM: MOV #7,R1   ; LOAD CODE
GO: BIS R1,R0   ; OR CODE WITH BITMASK
 MOV R0,2(R3) ; LOAD MTC
WAITCU: BIT #200,2(R3) ; CONTROLLER READY?
 BEQ WAITCU   ; NO, LOOP
 TST 2(R3)   ; ERROR?
 BMI ERR3   ; YES, ERROR TYPE 3
 CLR @4(R5)   ; IEOT=0
EXIT: CLR @(R5)+   ; IERR=0
 CLR @(R5)   ; IERREG=0
 RTS PC
ERR1: MOV #1,@2(R5) ; IERR=1
 CLR @4(R5)   ; CLEAR IERREG
 RTS PC
ERR2: MOV #2,@(R5)+ ; IERR=2
 MOV R1,@(R5) ; IFUNC TO IERREG
 RTS PC
ERR3: BIT #2000,(R3) ; EOT?
 BNE EOT
 BIT #40000,(R3) ; EOF?
 BNE EOF
 MOV #3,@(R5)+ ; IERR=3
 MOV (R3),@(R5)+ ; MTS TO IERREG
```

```
 CLR @(R5)   ; IEOT=0
 RTS PC
EOF: MOV #-1,@4(R5) ; IEOT=-1
 BR EXIT
EOT: MOV #1,@4(R5) ; IEOT=+1
 BIS #1400,2(R3) ; CLEAR FORMATTER EOT LATCH USING INVALID DRIVE
 BR EXIT
 .END
```

SORT PROGRAM

1. SORT PROGRAM DESCRIPTION

2. FORMAT OF PROGRAM DATA FILE - DK6:SORT.DAT

3. SORT LOG FILE

4. HOW HEADER BLOCKS ARE HANDLED BY THE SORT PROGRAM

5. SELECTING AND EDITING TRACES

6. EXAMPLE OF A 3-TRACE SORT, ILLUSTRATING HOW THE SORT ALGORITHM WORKS

7. CORRECTING FOR ERRORS IN SHIP SPEED AND SPLIT SORTING

8. DATA FORMAT TRANSLATION

1. SORT PROGRAM DESCRIPTION

The primary purpose of the SORT program is to re-order sets of seismic traces, so that data input from one tape in COMMON SHOT GATHERS are output to another tape in COMMON MID-POINT GATHERS. The SORT program accepts input gathers in either SEG-Y, DURHAM SEG-Y, or WESTERN'S SEG-Y tape formats. It outputs gathers in DURHAM SEG-Y only. A schematic diagram of the SORT program is given in fig. 1.

To carry out sorting, the program reads seismic traces from the input tape and stores them temporarily on disc until sufficient input data are available to assemble a group of output traces. The amount of disc space needed for temporary trace storage increases with the length of the seismic traces and with the square of the fold of cover. The SORT program uses the 256 MByte RM05 WINCHESTER disc for temporary trace storage and will handle data with traces up to 4096 samples and fold up to 48 channels.

The sorting algorithm used assumes that input data have been recorded with a shot spacing equal to half the receiver spacing and that there are no missing traces in a line. (There can, however, be blank traces, such as those inserted by the SEG-A DEMULTIPLEX program to cope with data lost during tape changes when recording data with only one tape drive). These restrictions make the Durham SORT program suitable only for marine data.

The SORT program will accept input COMMON SHOT GATHERS assembled either with the first trace of each gather originating from the nearest hydrophone and so on (Durham SDS 10/10 data), or with the first trace of each gather originating from the furthest hydrophone and so on (Western's and N.C.B. data).

The COMMON MID-POINT GATHERS output by the SORT program are arranged with the first trace of each gather being the one with least move-out, up to the last trace of the gather which has the largest move-out. (This is the arrangement assumed by all subsequent Durham processing programs).

Several secondary facilities for manipulating gathers and traces are included in the SORT program:-

(i) The fold of output gathers can be reduced from the fold of input gathers by sorting only a selected sub-set of the traces in each input gather.

(ii) Particular channels can be edited from input gathers and replaced by blank traces.

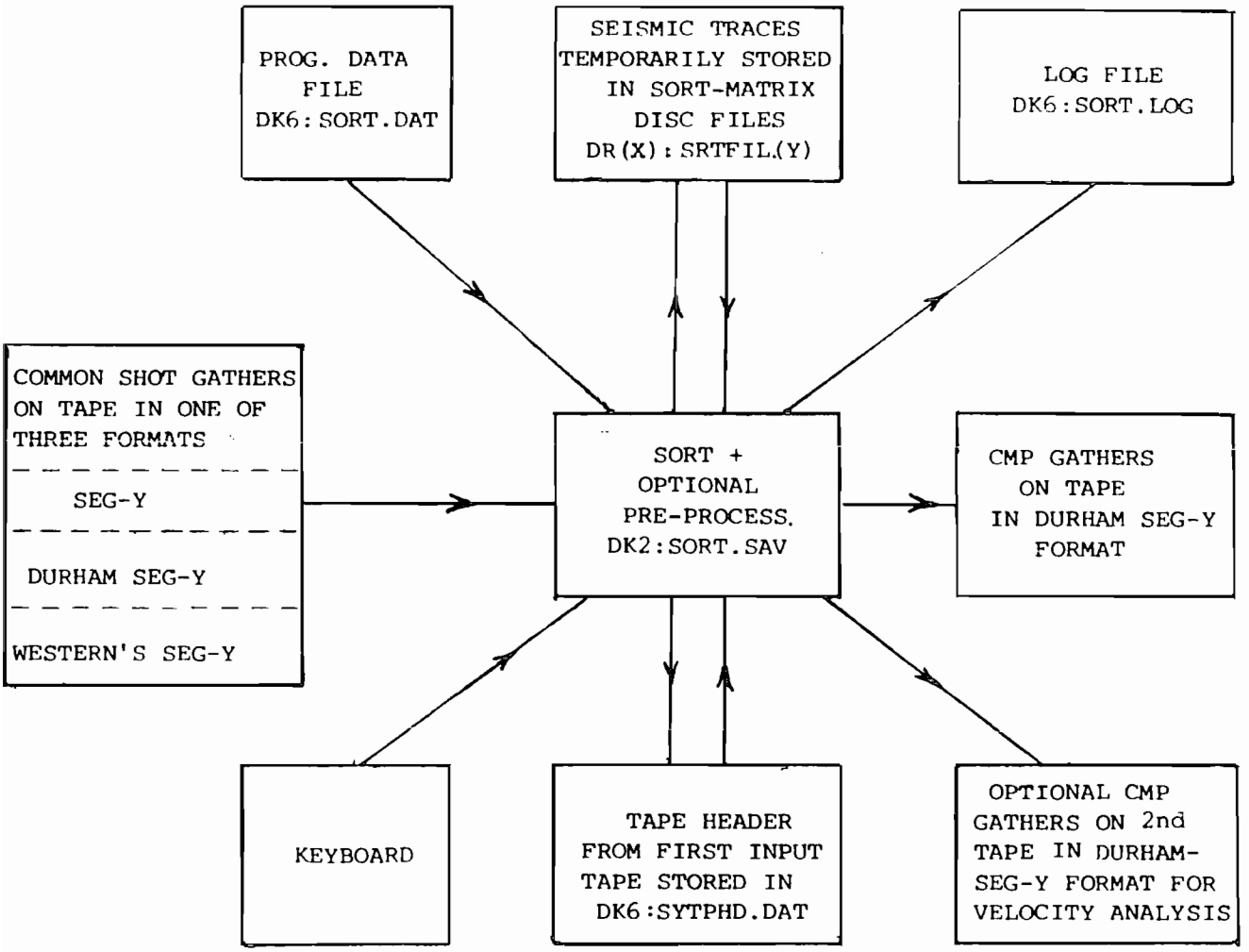


FIG. I. SORT PROGRAM SCHEMATIC

(iii) Each input trace can be inspected for the presence of noise spikes and the trace replaced by a blank trace if too noisy.

(iv) The sampling frequency of output traces can be reduced from that of input traces.

(v) Output traces can be "time-windowed" so that the start and end times of traces in output gathers need not be the same as those of traces in input gathers.

The SORT program includes facilities for selecting particular output gathers and writing them to a second output tape (in DURHAM SEG-Y format) for subsequent velocity analysis.

2. FORMAT OF PROGRAM DATA FILE DK6:SORT.DAT

NFILES, NTRACI, NTRACO, NROW, NREV, USEFLG, NFORM, NBLKTY, NSPLIT

NSMPIN, FSAMP, ITSTRT, ITFIN, NDEC

INDEX(1), INDEX(2), ..., INDEX(NROW)

ITRACO(1), ITRACO(2),...., ITRACO(NTRACO*NSPLIT)

IFILE(1), IFILE(2), ..., IFILE (NTRACO*NSPLIT)

} 12 values per line

NTRED, NEDTR(1), NEDTR(2),..., NEDTR(NTRED) IF NTRED > 0

NSPK, TRESH IF (NSPK ≥ 0)

VELNUM, (NVSRT, NVINT IF (VELNUM < 0)

or

VELAN(1), ..., VELAN(VELNUM) IF 0 < VELNUM ≤ 100)

TPDRR, TPDRW, TPDRV IF VELNUM ≠ 0

NDISK, IDISK(1), IDISK(2), ..., IDISK(NDISK)

NPROC, ((PROC(I,J); I=1,64), J=1,NPROC) IF NPROC > 0

NFILES (I6) Number of input COMMON SHOT GATHERS to sort.

The program terminates when this number of input gathers have been sorted. (When sorting a whole line of data, NFILES can be set larger than the number of input files available and the program terminated from the keyboard after the end of the last tape).

NTRACI (I6) Number of traces in each input COMMON SHOT GATHER (i.e. number of traces per input file).

NTRACO (I6) Number of traces in each output COMMON MID-POINT GATHER (i.e. number of traces per output file).

Normally NTRACE = NTRACI, so that all the traces of the input gathers take part in the sort process. However, NTRACO can be less than NTRACI when sorting a sub-set of the traces in each input gather.

NROW (I6) Number of rows in the sort file matrix.

Normally NROW = NTRACO and the sort matrix is square. However, NROW can differ from NTRACO when correcting errors due to wrongly measured ship speed (see sec. 7) or when using the SORT program to carry out certain types of data format translation (see sec. 8).

NREV (I6) Specifies the ordering of the traces within the COMMON SHOT GATHERS input to the SORT program:-

= 0 if the first trace of each input gather is from the near hydrophone (i.e. no trace reversal between input and output).
= 1 if the first trace of each input gather is from the far hydrophone (i.e. traces reversed between input and output).

Trace ordering within demultiplexed COMMON SHOT GATHERS input to the SORT program can vary. Data demultiplexed at Durham has the first trace within each COMMON SHOT GATHER originating from the nearest hydrophone. Other demultiplex data (e.g. Western's and N.C.B.) is assembled with the far hydrophone trace appearing first in the COMMON SHOT GATHER.

COMMON MID-POINT GATHERS output by the SORT program are, however, always arranged with the first trace of each gather being the one with least move-out, up to the last trace of the gather, which has the largest move-out.

The SORT program needs to know the trace ordering within the input files, so that the traces in the output gathers are ordered correctly.

USEFLG (I6) Specifies program re-start:-

= 0 for a new run (i.e. starting at beginning of a line).
= 1 for re-start, using SORT-MATRIX files already on disc.

If sorting is stopped part way through a line of data and subsequently re-started, the SORT program must be informed because traces stored temporarily on disc before the interruption of the run will be used when sorting resumes.

NFORM (I6) Specifies format of input data:-

= 0 for SEG-Y tape format
= 1 for DURHAM SEG-Y tape format
= 4 for WESTERN'S SEG-Y tape format

NBLKTY (I6) Controls type-out of tape header at start of program:-

= 0 to suppress header type-out
= 1 to display header

Setting this parameter to 1 displays on the terminal the ASCII tape header immediately before it is written to the first output tape in a line of data. This can often be a useful check of program operation.

NSPLIT (I6) Number of output CMP gathers produced from each input shot gather.
If shot spacing = 1/2 receiver spacing NSPLIT=1
If shot spacing = receiver spacing NSPLIT = 2
If shot spacing = 2 * receiver spacing NSPLIT = 4
etc.
See section 7(c) for example of use.

* * *

NSMPIN (I6) Number of samples in each input trace.

FSAMP (F10.0) Sampling frequency of input traces in kHz.

ITSTRT (I6) Absolute time (time from shot) to start of each output trace, in millisecs.

ITFIN (I6) Absolute time (time from shot) to end of each output trace, in millisec.

NDEC (I6) Decimation factor - the factor by which the sampling frequency of output traces is reduced from the sampling frequency of input traces.

= 1 (no reduction in sampling frequency)
= 2 (every second input sample used)
= 4 (every fourth input sample used)
= 8 (every eighth input sample used)

The above five parameters determine the relation between the length of traces in input gathers and the length of traces in output gathers.

ITSTRT and ITFIN are used to define a time-window which is filled by each output trace. If the time-window is shorter than the input traces, unwanted sections of the input traces are discarded. If the time window extends byeond the bounds of the input traces, the output traces are padded with zeros as necessary. In general, output traces will not have the same start time as input traces, so the SORT program updates the value of IDELAY (bytes 109-110) in the output trace headers to give the delay time (in millisec.) to the start of the output traces.

When sorting a line of data which has a recording delay that varies along the line, the time-window facility can be used to remove the variation. By defining a suitable window, all output traces can be given the same start and end times without loss of data. Traces with the greatest recording delay will be padded with zeros at their starts: short traces will be padded at their ends.

NDEC can be used to reduce the sampling frequency of output traces by factors of 2, 4 or 8. The SORT program picks every second, fourth or eighth sample from each input trace, depending on the value of NDEC. Intermediate samples are discarded. Before this operation, however, input traces are low-pass filtered to avoid aliasing the data. The values of ITIME in the output trace headers (bytes 117-118) are updated to give the new sampling frequency (in sec) of the decimated output data.

* * *

INDEX (1-NROW) (12I6)

Determines the order in which rows of the SORT MATRIX are filled. Normally, when sorting commences at the start of a line of data, INDEX is set to 1, 2, 3, ..., NROW. However, if the program is stopped part way through a line and then re-started, the values of INDEX must be cycled to correspond to the order for the last input gather that was processed before the program interruption. The log file contains the INDEX values used for the last gather sorted. See section 6 for more details about the sorting algorithm.

* * *

ITRACO (1-NTRACO*NSPLIT) (12I6)

Determines which of the traces in each input COMMON SHOT GATHER participate in the sorting operation. For the usual case where all the input traces take part, ITRACO is set to 1, 2, 3, ..., NTRACO. However, to reduce the fold of cover of the ouput gathers, sorting can be conferred to a sub-set of the traces in each input gather. Specify the trace numbers within an input gather of the required traces only. The values of ITRACO can be entered in any order.

* * *

IFILE(1-NTRACO*NSPLIT) (12I6)

Determines which group of COMMON SHOT GATHERS are to provide the traces from which an output COMMON MID-POINT GATHER is assembled. Normally IFILE is set to 1, 2, 3, ..., NTRACO. However, different IFILE values can be set when correcting errors due to wrongly measured ship speed (see sec. 8), or when using the SORT program to carry out certain types of format translations (see sec. 6).

* * *

NTRED (I6) Number of input traces to be edited out of each input gather (if none, enter 0). Edited traces are replaced by blank data.

NEDTR (1-NTRED) (12I6)

Determines which of the traces in each input gather are edited out and replace by blank traces. Specify the trace numbers within an input gather of the traces to be blanked. Values for NEDTR must be entered if NTRED $\neq$ 0.

The above two parameters can be used to replace selected traces in each input gather by blank data. This facility is useful, for example, for removing data recorded on malfunctioning channels. See sec. 5 for more details.

* * *

NSPK (I6) Controls "noise-spike" checking:-

$<$ 0 for no noise-spike checking.
$\geqslant$ 0 for the number of spikes tolerated before the trace is blanked.

THRESH (F10.0) Used to specify the D.C. offset above or below which signal is regarded by the "de-spiking routine" as an unwanted noise spike. A value for THRESH must be entered if NSPK $\geqslant$ 0.

The above two parameters control a routine which can be used to inspect each input data trace for the presence of noise spikes and replace the trace with blank data if it is too noisy. Setting NSPK to a positive value specifies the number of noise spikes tolerated before the whole trace is blanked.

Note that if more than a quarter of the input data samples are found to be noise spikes, the whole output trace is blanked regardless of the value of NSPK (providing NSPK $>$ 0).

In the case where trace-windowing is used (see ITSTRT and ITFIN above) only the region of interest is scanned for the presence of noise spikes.

If NSPK is negative, no "spike-checking" is performed and no traces will be blanked, however noisy they are.

* * *

VELNUM (I5) Determines how velocity analysis gathers are selected:-

= 0 for no output of velocity gathers.
= -1 for velocity gathers picked off at regular intervals along a line of data until program termination (subject to a maximum of 100 velocity gathers).
= > 0 to indicate a number of velocity gathers (max. 100) that will be individually specified.

NVSRT (I6)
NVINT (I6) } Used when velocity analysis gathers are picked-off at regular intervals along a line of data, (i.e. when VELNUM = -1). NVSRT specifies the CMP number of the first gather to be output to the velocity tape. NVINT specifies the interval at which CMP gathers are picked-off.

If VELNUM ≠ -1, do not enter values for these two parameters.

VELAN (1-VELNUM) (12I6)

Used when explicitly specifying which CMP gathers are to be output as velocity analysis gathers (i.e. when VELNUM > 0). Enter the CMP numbers of the gathers required.

If VELNUM ≤ 0 do not enter values for VELAN.

The above four parameters determine which CMP gathers are written to a second output tape for subsequent velocity analysis. There are two methods of selecting velocity gathers.

(a) To pick off velocity gathers at regular intervals along a line, set VELNUMN = -1. Use NVSRT to specify the CMP number of the first gather to write to the velocity tape, and NVINT for the number of CMP gathers from one velocity output to the next. Do not set values for VELAN.

(b) To explicitly specify which CMP gathers are to be output to the velocity tape, set VELNUM equal to the total number of velocity gathers to be written, and use VELAN to indicate the CMP numbers of the gathers required. Do not set values for NVSRT or NVINT.

(c) If no velocity output is required, set VELNUM = 0. In this case, values for VELAN, NVSRT or NVINT need not be given.

Note that which ever method for selecting velocity gathers is used, there is no need to update the values of NVSRT, NVINT or VELAN if sorting is stopped part way through a line of data and subsequently re-started.

Unless VELNUM = 0 (no velocity gathers output), a value for TPDRV, the velocity tape drive number must be given (see below).

* * *

TPDRR (I6) Input tape drive number (0, 1 or 2).

TPDRW (I6) Output tape drive number (0, 1 or 2).

TPDRV (I6) Velocity tape drive number (0, 1 or 2). If VELNUM = 0 above (see velocity gather output), there is no need to give a value for this parameter.

* * *

NDISK (I6) Number of "DR" RM05 discs available for temporary storage of seismic traces (max. 4).

IDISK (1-NDISK) (12I6)

Specifies the DR numbers of the discs available for storing seismic traces. (Numbers in the range 0-7).

The above two parameters determine which discs the SORT program may use for the creation of SORT-MATRIX FILES, in which successive input traces are temporarily stored during the sorting operation.

Before sorting data, it is good practice to check that there is plenty of free space on the discs specified, and to squeeze the discs to ensure that the free space is not fragmented into regions which are too small to accommodate SORT-MATRIX FILES.

* * *

NPROC (I6) Number of lines of ASCII text to add to the tape header block on each ouput tape (Max. 4).

PROC (I,J) I=1,64; J-1,NPROC (64A1)

Enter lines of ASCII text to add to the output tape header blocks. Each line is up to 64 characters long. (If NPROC = 0, no text need be entered).

Using the above two parameters, up to 4 lines of text can be added to the ASCII header block of each output tape; for example to describe operations that can be performed at the same time as sorting, such as spike checking, sample windowing and sampling frequency reduction. The specified text is added in lines 25-28, inclusive, of the header block. If NREV = 1 (see above), the message

ORIGINAL TRACE ORDERING REVERSED

is automatically added after any text that the user enters.

The ASCII tape header that is written to the first output tape can be displayed on the terminal (as a check of program operation) by setting NBLKTY = 1 (see above).

3. SORT LOG FILE

The log file produced by the SORT program provides a record of tape changes and lists errors detected by the "de-spiking" routine, if this is selected. It also gives a status report on program termination. The log file can be displayed on the terminal, or printed, after the SORT program has stopped.

If "de-spiking" is selected (NKPK >,0) an error report is produced for each input COMMON SHOT GATHER sorted. The original field record number of the gather is recorded, followed by the number of spikes detected in each trace of the gather. Traces that have been blanked by the "de-spiking" routine because more than NSPK noise spikes were found, are marked with a '*' character. Likewise, a '*' is used to mark traces that were already blank on input to the SORT program, or that have been edited out during sorting.

No error report is produced if "de-spiking" is not selected (NSPK < 0).

At each change of input or output tape, the current CMP output gather number is given.

On termination of the SORT program, the log file records the trace sequence number of the last trace written to the output tape; the last INDEX values; and the CMP number of the last gather output. These "terminal" values are needed if the SORT program is re-started to continue sorting more of the same line of data. In these circumstances, update the program data file by changing the INDEX values to those shown at the end of the log file. Also set the value of USEFLG to 1. Add one to the last output trace sequence number to determine the value requested interactively at the terminal when the SORT program is re-started.

4 HOW HEADER BLOCKS ARE HANDLED BY THE SORT PROGRAM

4(a) Tape Header Blocks

The SORT program handles tape header blocks by storing a copy of the headers from the first input tape in a permanent disc file named DK6:SYTPHD.DAT. Each time a new output tape is started, a copy of the headers is take from the disc file; certain entries in the headers are updated by the program, and the new headers are written to the output tape.

A fresh copy of the tape headers is placed in DK6:SYTPHD.DAT each time the SORT program is started, provided that the input tape is at BOT. At the start of sorting a line of data, this is always the case and a copy of the tape headers is, therefore, always available for the program. If the SORT program is stopped and re-started part-way through a line, a fresh copy of the tape headers is taken if the program re-start happens to correspond with the start of a new input tape: if not, the program uses the older version of the tape headers that was stored in DK6:SYTPHD.DAT before the interruption of the run.

The entries in the ASCII tape header block that are updated before writing the tape headers to a new output tape are:-

REEL. NO.
DATA TRACES/RECORD
CMP-FOLD
SAMPLE INTERVAL (μsec)
SAMPLES/TRACE

Also any text entered in PROC in the program data file is added in lines 25, 26, 27 and 28 of the ASCII header.

The entries in the binary tape header block that are updated before writing the tape headers to a new output tape are:-

REEL NO.	(bytes 9-12)
NO. OF DATA TRACES/RECORD	(bytes 13-14)
SAMPLE INTERVAL (s)	(bytes 17-18)
NO. OF SAMPLES/DATA TRACE	(bytes 21-22)
CMP FOLD	(bytes 27-28)
TRACE SORTING CODE (set = 2)	(bytes 29-30)

4(b) Trace Header Blocks

The trace header attached to every trace on the input tape is interrogated and updated by the SORT program before being output.

In order for the SORT program to work properly, the following entries must be present in the trace headers on the input tape:

ORIGINAL FIELD RECORD NO.	(bytes 9-12)	JFNUM
TRACE NO. IN ORIGINAL FIELD RECORD	(bytes 13-16)	JTRAS -See Note 1
TRACE IDENTIFICATION CODE	(bytes 29-30)	ITYPE -See Note 2
RECORDING DELAY (in millisec)	(bytes 109-110)	IDELAY-See Note 3

Note 1. If JTRAS is not present in the input trace headers, the SORT program still runs, but JTRAS is set = 1 in the trace headers of all the output traces. (This action enables the SORT program to carry out formation translation of data already sorted into CMP gathers, but with JTRAS omitted from the trace headers - e.g. data sorted by LDGO).

Note 2. If ITYPE is not present in the input trace headers, the SORT program still runs, but sections of program code that would normally be skipped on encountering "dead" or "dummy" traces will still be executed. This will needlessly increase program run time.

Note 3. If IDELAY is not present in the input trace headers, the SORT program still runs, but the values of IDELAY entered in the output trace headers may be incorrect. This means that subsequent processing and velocity analysis programs may not work.

The following entries in the output trace headers are generated by the SORT program:-

CMP GATHER NO.	(bytes 21-24)	JCDPNO
TRACE NO. IN CMP GATHER	(bytes 25-28)	JCDPTR
RECORDING DELAY (millisec)	(bytes 109-110)	IDELAY
NO. OF SAMPLES/TRACE	(bytes 115-116)	ISAMP
SAMPLE INTERVAL in μs	(bytes 117-118)	ITIME

The following trace header entry is set if a trace is edited or blanked by the SORT program:

TRACE IDENTITY CODE	(bytes 29-30)	ITYPE	(set = 2)

The entries JFNUM (bytes 9-12) and JTRAS (bytes 13-16) mentioned above, although used by the SORT program, are not recalculated but passed directly to the output trace headers. (The exception is when JTRAS is not entered in the input trace headers, in which case, JTRAS is set to 1 in all the output trace headers).

Any other entries in the input trace headers are passed directly to the output trace headers.

5. SELECTING TRACES AND EDITING TRACES

A sub-set of the traces in each input COMMON SHOT GATHER can be selected and sorted by suitable choice of values of NTRACO and ITRACO in the data file. For example, to select and sort channels 1, 2, 3, 4, 5, 7, 8, 10, 11, 12 from 12 fold input data, set:-

NTRACI = 12

NTRACO = 10

NROW = 10

INDEX = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

ITRACO = 1, 2, 3, 4, 5, 7, 8, 10, 11, 12

IFILE = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

The program recognises the input traces required by decoding the "trace no. within the the original field record" from each input trace header (bytes 13-16) and comparing it with the values of ITRACO. (Note that if the "trace no. within the original field record" is not filled in for the input data, the program automatically sets this value to 1 for all input traces, and the trace selection facility cannot be used. This applies to some data sorted by LDGO).

In a similar way, particular input traces which are known to have been recorded on malfunctioning recording channels can be edited out by suitable choice of values for NTRED and NEDTR. Edited channels still participate in the sort algorithm so that the re-ordering of traces into CMP gathers on the output tape is not disturbed. However, an edited trace is blanked before it enters the sorting procedure and the value of the "trace identification code" in its trace header (bytes 29-30) is set to 2 to indicate a "dead" trace.

For example to edit out traces 6 and 9 from a 12 channel sort, set:-

NTRACI = 12

NTRACO = 12

NROW = 12

NTRED = 12

INDEX = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

ITRACO = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

IFILE = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

NEDTR = 6, 9

The program decodes the "trace no. within the original field record" from the trace header of each of the input traces (bytes 13-16) and compares it with the values of NEDTR to determine which traces to blank. (Note that if the "trace no. within the original field record" is not filled in for the input data, the program automatically sets this value to 1 for all input traces, and the trace editing facility cannot be used).

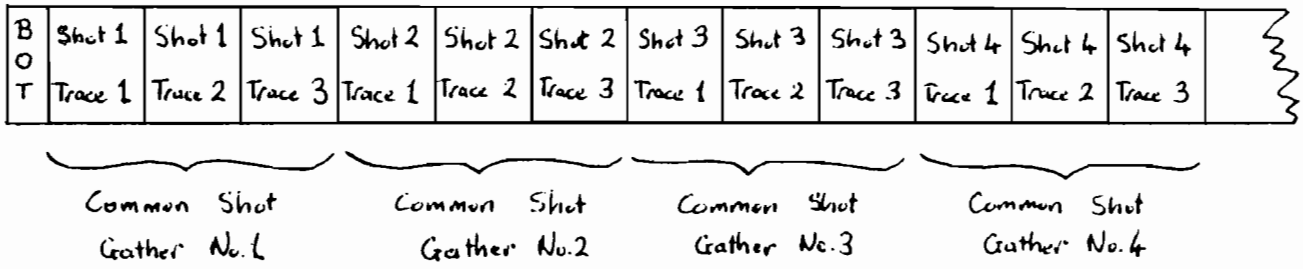
Notice the difference between selecting ten traces from 12 fold input data and editing out two traces from the same 12 fold input data. In the first case, CMP gathers of 10 traces each are written to the output tape. In the second case, CMP gathers of 12 traces each are output, but with the 6th and 9th traces blanked.

Whether to select traces or to edit traces from input data depends on preserving the normal move-out relation of the traces in the output CMP gathers. Selecting traces is appropriate where hydrophones have been connected only to some of the recording channels. (The unused recording channels are still recorded, even though no hydrophone is connected). Editing is appropriate where hydrophones have been connected to all the recording channels, but some of the recording channels have malfunctioned.

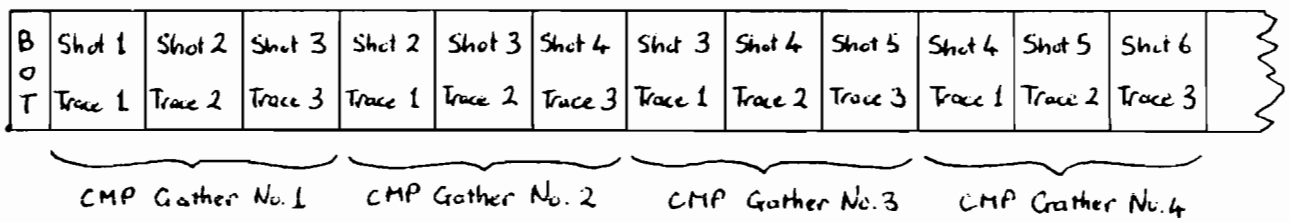
Finally, note that traces which are blanked by the "de-spiking routine" (NSPK ≥ 0 in the program data file) still take part in the sort operation and are still written to the output tape, so that the correct normal move-out relation is preserved in output CMP gathers. (Traces blanked because they contain too many noise spikes have the "trace identification code" in their trace headers (bytes 29-30) set to 2 to indicate a "dead" trace).

6. EXAMPLE OF A 3-TRACE SORT, ILLUSTRATING HOW THE SORT ALGORITHM WORKS

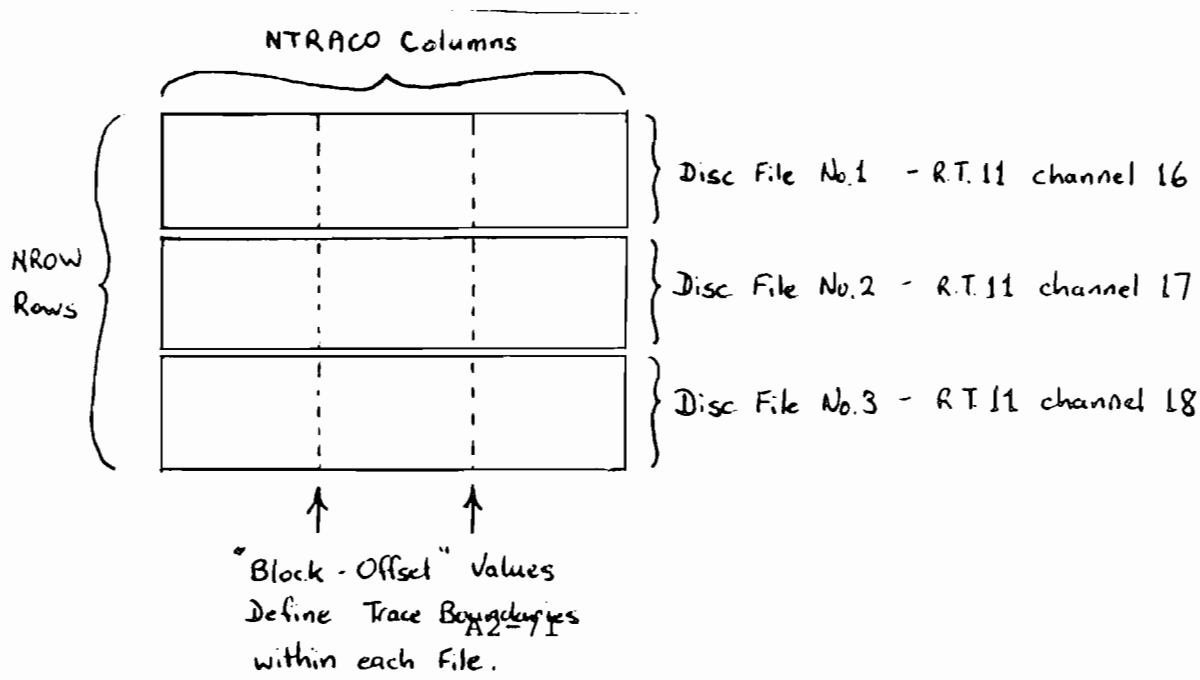
Traces on a demultiplexed input tape are arranged in COMMON SHOT GATHERS:-



Traces on the SORTED output tape are arranged in CMP GATHERS:-

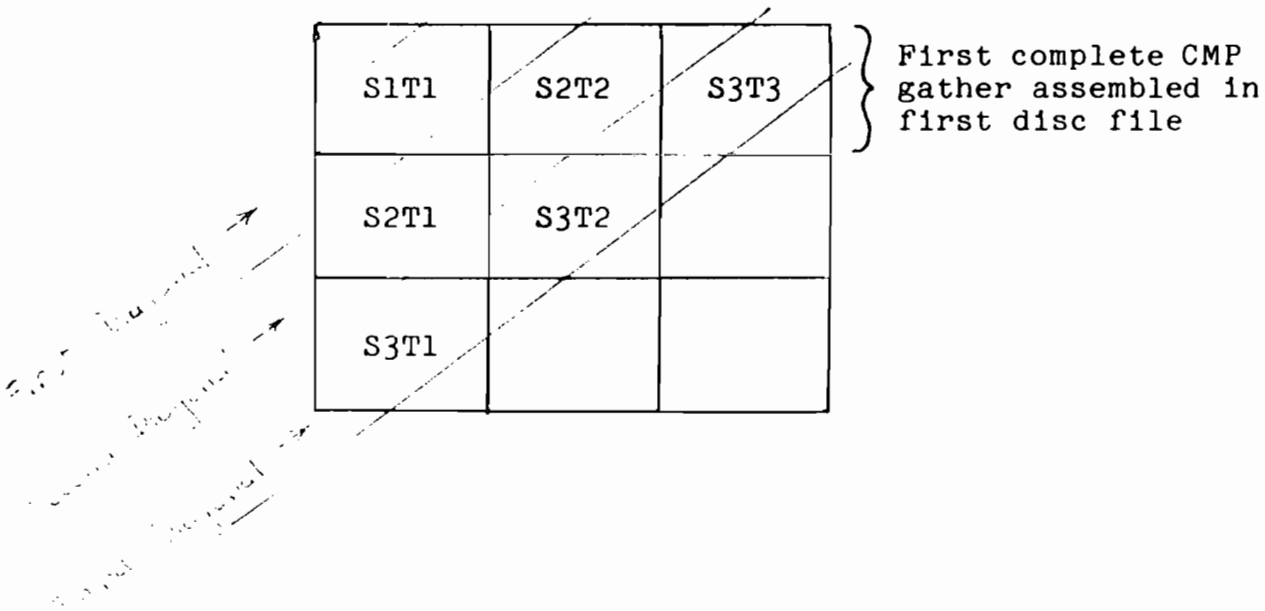


The sort program creates files on disc which act as temporary storage for traces read from the input tape. The SORT operation is best understood by regarding these disc files as a 2-dimensional matrix for trace storage:-



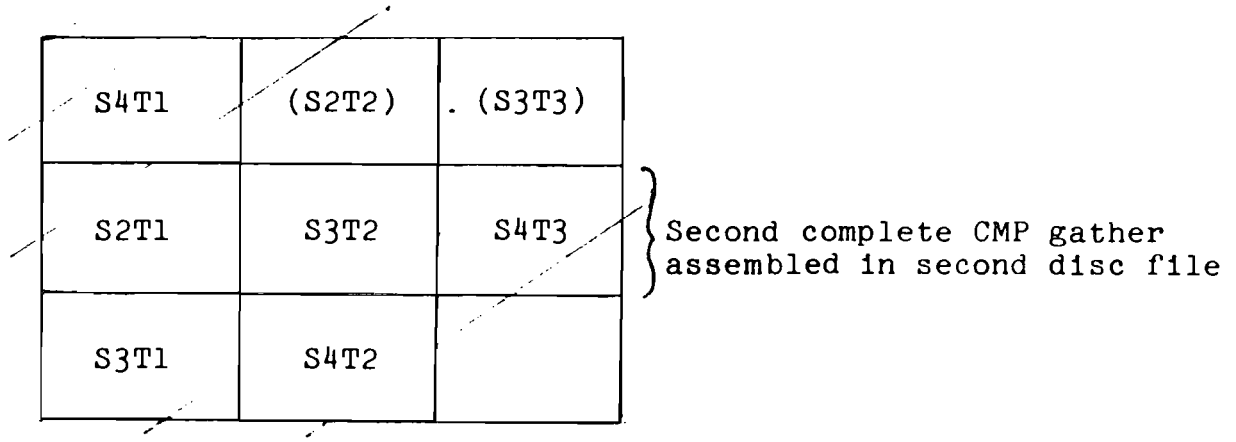
For a normal SORT (i.e. no correction for ship speed) NROW equals NTRACO and the matrix is square. Each disc file is of sufficient length to hold NTRACO traces, with each trace rounded up to a block boundary by padding with zeros (the zeros are not output to the CMP gather tape).

Successive input traces are placed in the matrix so that eventually each file (horizontal row in the matrix) contains a CMP gather, which can then be output to tape. The matrix fills along diagonals, starting at the top left corner of the matrix and filling each diagonal from bottom left to top right. After the first 3 COMMON SHOT GATHERS have been read from the input tape, the matrix contains the following traces:-



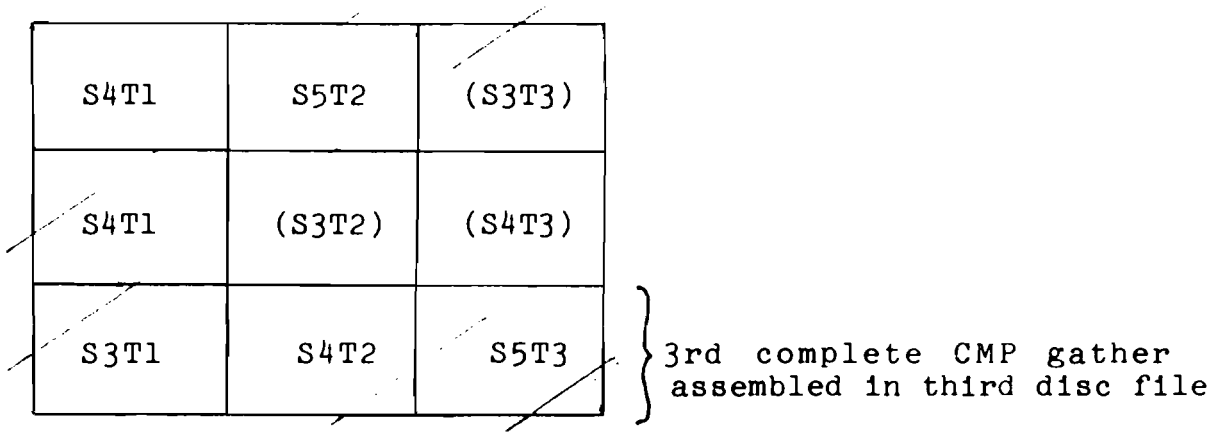
Note that only one trace from the first COMMON SHOT GATHER is used, and only two from the second. But there are spaces in the matrix for all the traces from the third COMMON SHOT GATHER. (In general, if each input file contains n traces, the SORT-MATRIX for a normal SORT is n x n, and nth input file is the first to have all its traces put into the matrix).

As complete CMP gathers assemble in the SORT FILE MATRIX they are ouput to tape, leaving the disc files occupied by the output traces free to be overwritten by new incoming data. After the fourth COMMON SHOT GATHER has been read from the input tape, the matrix contains the following traces:-



Traces in brackets have been output as part of a CMP gather, but have not yet been overwritten by new input data.

After the fifth COMMON SHOT GATHER is input, the matrix contains:-



In the SORT program, the matrix of disc files is NTRACO columns wide by NROW rows deep and the position in the matrix into which each successive input trace is stored is controlled by the arrays INDEX and FILE.

INDEX determines the order in which rows of the SORT MATRIX are filled. At the start of a program run, the values of INDEX in the data file are set to the row numbers into which the first trace of successive COMMON SHOT GATHER inputs are to be placed. Thus for a normal, 3 trace sort, INDEX is set to 1, 2, 3 and (S1T1) is placed in the first row of the matrix, (S2T1) in the second row, and (S3T1) in the third row.

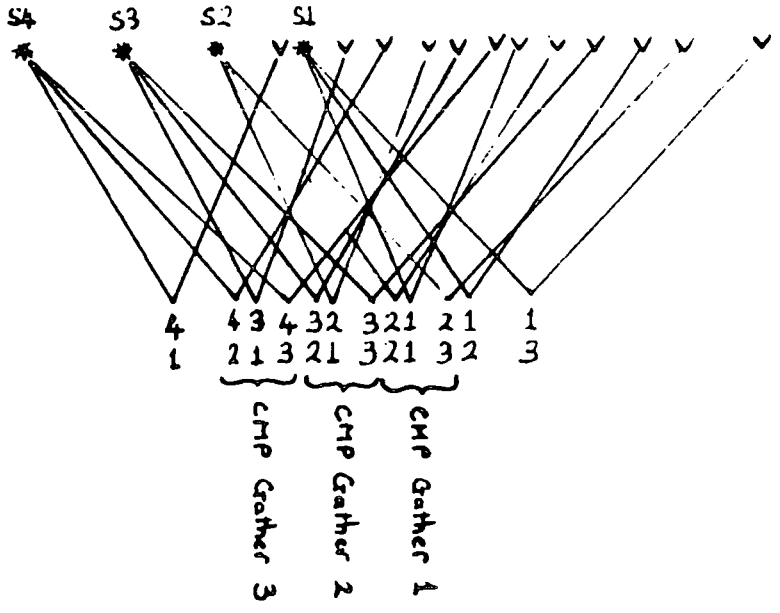
As the SORT program runs, the values of INDEX are cycled once for every COMMON SHOT GATHER input, so that the program always knows which row the next input trace is to be read into. If the SORT program is stopped part way through a line of data,

the values of INDEX in the data file must be cycled to correspond to the order for the last input file processed before continuing (the log file gives this information).

7. CORRECTING FOR ERRORS IN SHIP SPEED

Differences between a ship's water speed and land speed (caused by currents) cause errors because the traces in a conventional CMP gather will not share a single common depth point. Sometimes the situation can be improved by choosing a different set of traces for the gather. In the SORT program this can be achieved by choosing suitable values for NROW and IFILE. The following examples illustrate the technique for the case of a 3-trace sort.

7(a) Fast ship (i.e. Steaming with current)



In the above diagram it can be seen that a fast ship can result in the best approximation to ideal CMP gathers being:-

(S1T1)(S2T2)(S2T3) — gather 1

(S2T1)(S3T2)(S3T3) — gather 2

(S3T1)(S4T2)(S4T3) — gather 3

N.B. The output CMP spacing is proportional to the aquisition shot spacing.

To achieve this CMP ensemble pattern, choose the following values in the data file:-

NTRACO = 3

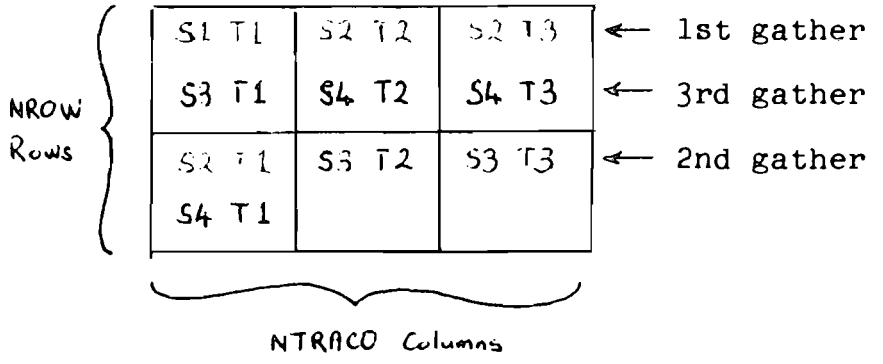
NROW = 2

INDEX = 1, 2

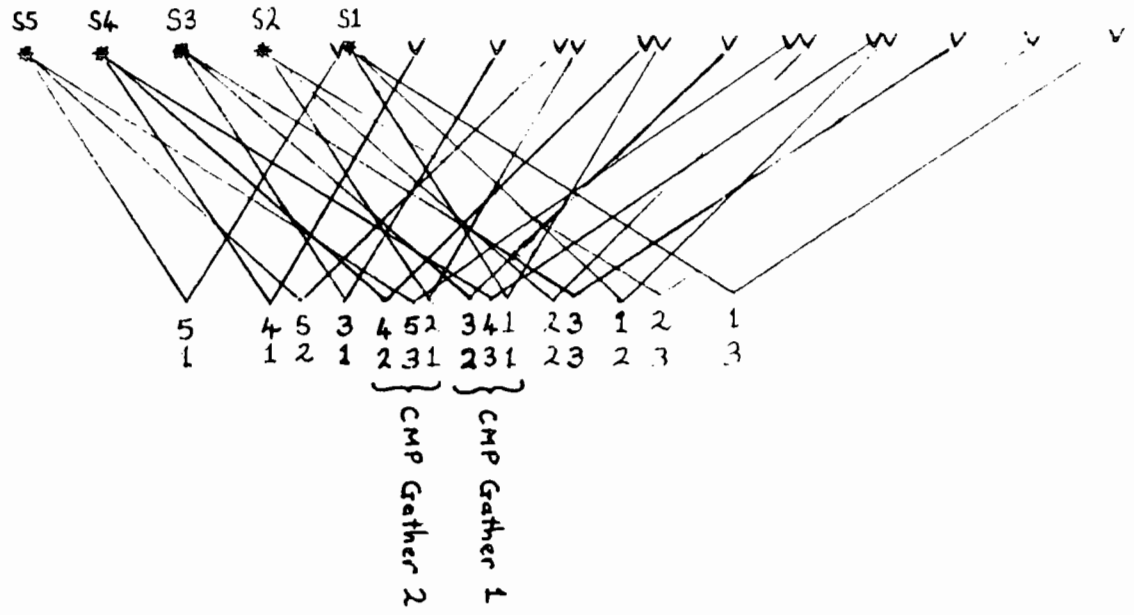
ITRACO = 1, 2, 3

IFILE = 1, 2, 2

The SORT MATRIX in this case will not be square, and will be filled in the following order:-



7(b) Slow ship (i.e. Steaming against current)



In the above slow ship example, the best approximation to perfect CMP gathers is:-

(S1T1)(S3T2)(S4T3) (S2T1)(S4T2)(S5T3) (S3T1)(S5T2)(S6T3) ...

gather 1 gather 2 gather 3

... (S4T1)(S6T2)(S7T3) (S5T1)(S7T2)(S8T3)

gather 4 gather 5

To achieve this CMP ensemble pattern, chose the following values in the data file:-

NTRACO = 3

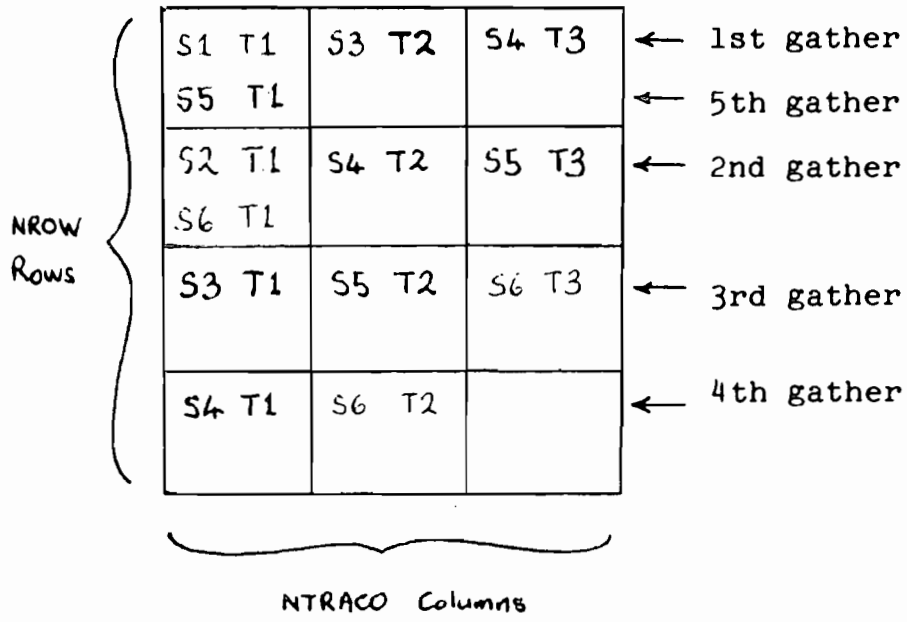
NROW = 4

INDEX = 1, 2, 3, 4

ITRACO = 1, 2, 3

IFILE = 1, 3, 4

The SORT MATRIX will not be square and will fill in the following order:-



7(c) Split Sort

Two examples are given for the split sort algorithm; for the case where the shot interval is equal to the receiver spacing; and for the case where the shot interval is twice the receiver spacing.

Example 1.

For eight fold common shot gather data.

```
NTRACO = 4
NTRACI = 8
NSPLIT = 2
INDEX  = 1, 2, 3, 4, 5, 6, 7, 8
ITRACO = 1, 2, 3, 4, 5, 6, 7, 8
IFILE  = 2, 1, 4, 3, 6, 5, 8, 7
```

N.B. The order of IFILE to get the greater offset CMP output first onto the tape.

The output CMP gathers will contain the following traces

```
(S1T2),(S2T4),(S3T6),(S4T8)   (S1T1),(S2T3),(S3T5),(S4T7)

        gather 1                        gather 2


(S2T2),(S3T4),(S4T6),(S5T8)   ...

        gather 3
```

Example 2.

For eight fold common shot gather data.

```
NTRACO = 2
NTRACI = 8
NSPLIT = 4
INDEX  = 1, 2, 3, 4, 5, 6, 7, 8
ITRACO = 1, 2, 3, 4, 5, 6, 7, 8
IFILE  = 4, 3, 2, 1, 8, 7, 6, 5
```

The output CMP gathers will contain the following traces

```
(S1T4),(S2T8)   (S1T3),(S2T7)   (S1T2),(S2T6)   (S1T1),(S2T5)

  gather 1        gather 2        gather 3        gather 4


(S2T4),(S3T8)   (S2T3),(S3T7)   ...

  gather 3        gather 4
```

8. DATA FORMAT TRANSLATION

The SORT program can be used as a means of translating data in SEG-Y or WESTERN'S SEG-Y tape formats to DURHAM SEG-Y tape format. Translation is achieved by choosing suitable values for NROW, INDEX, ITRACI, ITRACO, IFILE in the program data file.

To translate single channel data (e.g. stacked data in SEG-Y to DURHAM SEG-Y) set:-

```
NTRACI  = 1
NTRACO  = 1
NROW    = 1
INDEX   = 1
ITRACO  = 1
IFILE   = 1
```

} Single Trace Sort

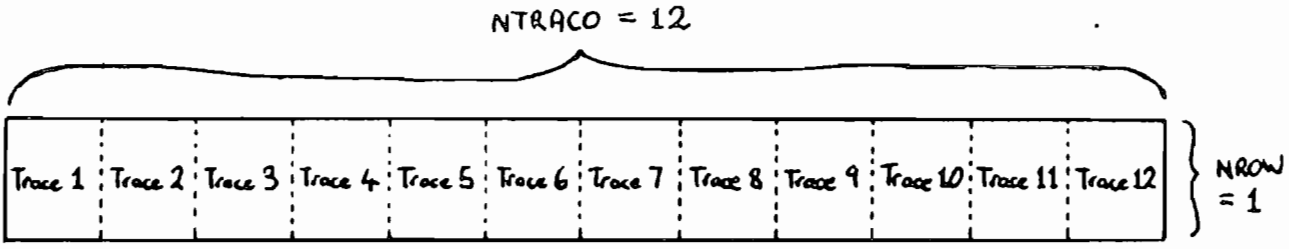
With these values set, no SORT MATRIX FILE is opened because only one trace is being translated at a time.

To translate sorted data of, say, 12 channels, create a data file with:-

```
NTRACI  = 12
NTRACO  = 12
NROW    = 1
INDEX   = 1
ITRACO  = 1,2,3,4,5,6,7,8,9,10,11,12
IFILE   = 1,1,1,1,1,1,1,1,1,1,1,1
```

} Single File Sort

With these values set, one SORT MATRIX file is opened on disc and it is of sufficient length to hold all twelve traces of each input file:-



SORT MATRIX FILE FOR SINGLE FILE SORT (12 CHANNEL)

The traces of the input file are read successively into the SORT MATRIX file on disc and then all twelve traces are output as one CMP gather. The order of the traces on the output tape is thus the same as the order on the input tape.

Note that the single file sort is effected by regarding the 12 trace CMP gathers on the input tape as 12 traces of a COMMON SHOT GATHER and setting all the IFILE values to 1. The means that each column of the SORT MATRIX is filled with a trace from the same input file.

```
C
C  GENERAL PURPOSE SORTING PROGRAM FOR
C  SEISMIC DATA FILES. BASED ON A SORT
C  MATRIX DEVISED BY M J POULTER OCT 80
C
C  MOD FEB 82 R W HOBBS
C  TO WORK WITH SYSTIME TAPE DECKS;TO GIVE CHANNEL EDITING THRO' LINE
C
C  MOD MAY 82 R W HOBBS
C  TO DECIMATE DATA BY INTEGER FRACTION;TO IDENTIFY AND REMOVE
C  SPIKES IN DATA
C
C  MOD NOV 82 R W HOBBS/D W WELLBY
C  TO READ/WRITE SEG-Y
C
C  MOD AUG 84 R W HOBBS
C  TO ENABLE SPLIT SORT. I.E. WHEN SHOT AND RECEIVER GROUPS HAVE SAME SPACING
C
      INTEGER*2 VELNUM,USEFLG,TPDRR,TPDRW,TPDRV,
     +VELAN(101),REELNU,INDEX(60),IPOS(48),
     +IEDTR(48),ITRO(48),IBLKOF(48),VNUM,VFIL,VFLAG,ITMERR(2)
      REAL*4 AWESTN
      REAL*8 FNAME,MTNAM(3)
      LOGICAL*1 PROC(64,4),OUTBUF(4156)
      COMMON /WEST/AWESTN
      COMMON /DATBUF/OUTBUF
      COMMON /OUTPUT/NTRACO,IBLKOF,IWORDW,FTRCNO,TPDRW,IEOTW,ISMPOT,NROW
      COMMON /VELOCY/TPDRV,IEOTV,VFIL,VELNUM,VFLAG
      COMMON /INPUT/IWORDR,TPDRR,IEOTR,NFILES,NTRACI,NSMPIN,NFORM,
     +INDEX,IPOS,VELAN,VNUM,ITSTRT,USEFLG,ISMPIN,NSPLIT
      COMMON /PROCES/NDEC,NSPK,THRESH,IEDTR,ITRO,FSAMP,ITIMOT,
     +NTAP,NTRANF,N3,N4,NREST
      COMMON /REELS/PROC,NPROC,NREV,NBLKTY
      COMMON /TAPE/MTNAM,ITMERR
      DATA MTNAM(1)/3RTM4/,MTNAM(2)/3RTM5/,MTNAM(3)/3RTM6/
      DATA APHAN/3RAP /,TAPEHN/3RTM /,DRHAN/3RDR /
C
C SET UP I/O
C
      IF(IFETCH(APHAN).NE.0)STOP'AP FETCH ERR'
      IF(IFETCH(TAPEHN).NE.0) STOP 'TAPE FETCH ERR'
      IF(IFETCH(DRHAN).NE.0) STOP 'DR FETCH ERR'
      IF(ICDFN(64).NE.0)STOP' CHANNEL SET ERROR'
      CALL APINIT(0,0,1)
C
C  CALL INPUT SUBROUTINE
C
      CALL SORTIN(REELNU)
      ISTOP=0
C
C  LOOKUP WRITE TAPE DRIVE
C
      CALL SORTHD(ICDPNO,ITIMOT,REELNU,TPDRW,IEOTW,ISTOP)
      IF(ISTOP.NE.0) CALL SRTEND(ISTOP,ICDPNO)
C
C  LOOKUP VELOCITY TAPE DRIVE IF SEG-Y OUTPUT
C
      IF(VELNUM.EQ.0) GO TO 150
      CALL SORTHD(ICDPNO,ITIMOT,VFLAG,TPDRV,IEOTV,ISTOP)
      IF(ISTOP.NE.0) CALL SRTEND(ISTOP,ICDPNO)
C
C  DO SORT
C
 150  CALL SRTSUB(ICDPNO,REELNU,ISTOP)
C
C CLOSE DOWN CODE
C
      CALL SRTEND(ISTOP,ICDPNO)
      CALL RESET
      CALL EXIT
      END
```

```
      SUBROUTINE SORTIN(REELNU)
C
C  INPUT SUBROUTINE TO THE SORT PROGRAM
C
C  READS INPUT DATA FILE ON UNIT 1
C  SETS UP CONSTANTS FOR THE REST OF THE PROGRAM
C  OPENS SORT FILES ON THE DISKS
C  LOOKS UP READ TAPE DRIVE, CHECKS INPUT FORMAT
C  WRITES ASCII AND BINARY TAPE HEADERS TO DISK FILE DK6:SYTPHD.DAT
C
      INTEGER*2 VELNUM,USEFLG,TPDRR,TPDRW,TPDRV,VFLAG,
     +VELAN(101),REELNU,IDUM(256),INDEX(60),ITRACO(48),IPOS(48),
     +IFILE(48),IEDTR(48),NEDTR(48),ITRO(48),IBLKOF(48),VNUM,VFIL,
     +ITMERR(2),IDISK(4)
      REAL*8 FNAME,MTNAM(3)
      REAL*4 AWESTN
      LOGICAL*1 PROC(64,4),SRTRT(14),
     +RLHDBL(80,40),RLHDCD(400),OUTBUF(4156)
      COMMON /WEST/AWESTN
      COMMON /DATBUF/OUTBUF
      COMMON /OUTPUT/NTRACO,IBLKOF,IWORDW,FTRCNO,TPDRW,IEOTW,ISMPOT,NROW
      COMMON /VELOCY/TPDRV,IEOTV,VFIL,VELNUM,VFLAG
      COMMON /INPUT/IWORDR,TPDRR,IEOTR,NFILES,NTRACI,NSMPIN,NFORM,
     +INDEX,IPOS,VELAN,VNUM,ITSTRT,USEFLG,ISMPIN,NSPLIT
      COMMON /PROCES/NDEC,NSPK,THRESH,IEDTR,ITRO,FSAMP,ITIMOT,
     +NTAP,NTRANF,N3,N4,NREST
      COMMON /REELS/PROC,NPROC,NREV,NBLKTY
      COMMON /TAPE/MTNAM,ITMERR
      EQUIVALENCE (OUTBUF(1),RLHDBL(1,1)),(OUTBUF(3201),RLHDCD(1)),
     +(OUTBUF(1),IDUM(1))
      DATA VELAN/101*0/
      DATA IDISK/4*9/
      DO 77 I=1,256
 77   IDUM(I)=0
C
C  READ IN DATA FILE DK6:SORT.DAT
C
      OPEN(UNIT=1,NAME='DK6:SORT.DAT',TYPE='OLD',ERR=200)
      GO TO 210
 200  STOP 'NO DATA FILE'
 210  DO 215 I=1,60
 215  INDEX(I)=1
      READ(1,1000) NFILES,NTRACI,NTRACO,NROW,NREV,USEFLG,NFORM,NBLKTY,
     +NSPLIT
      IF(NSPLIT.LE.0) NSPLIT=1
      IF(NROW.GT.60) STOP'MAX. 60 ROWS IN SORT MATRIX'
      IF(NTRACI.GT.48.OR.NTRACO*NSPLIT.GT.48) STOP'MAX. 48 CHAN SORT'
      READ(1,1100) NSMPIN,FSAMP,ITSTRT,ITFIN,NDEC
 1100 FORMAT(I6,F10.0,3I6)
      READ(1,1000) (INDEX(I),I=1,NROW)
      READ(1,1000) (ITRACO(I),I=1,NTRACO*NSPLIT)
      READ(1,1000) (IFILE(I),I=1,NTRACO*NSPLIT)
 1000 FORMAT(12I6)
      READ(1,1000) NTRED
      IF(NTRED.LE.0) GOTO 300
      IF(NTRED.GT.NTRACI) STOP'TOO MANY EDITED CHANS'
      BACKSPACE 1
      READ(1,1000) NTRED,(NEDTR(I),I=1,NTRED)
 300  READ(1,1000) NSPK
      IF(NSPK.LT.0) GOTO 310
      BACKSPACE 1
      READ(1,1100) NSPK,THRESH
 310  READ(1,1000) VELNUM
      IF(VELNUM.EQ.0) GOTO 320
      BACKSPACE 1
      IF(VELNUM.LT.0) READ(1,1000) VELNUM,NVSRT,NVINT
      IF(VELNUM.GT.100) STOP'TOO MANY VELOCITY FILES'
      IF(VELNUM.GT.0) READ(1,1000)VELNUM,(VELAN(I),I=1,VELNUM)
      READ(1,1000) TPDRR,TPDRW,TPDRV
      GOTO 330
 320  READ(1,1000) TPDRR,TPDRW
 330  READ(1,1000) NDISK
```

```
      BACKSPACE 1
      IF(NDISK.LT.1.OR.NDISK.GT.4) STOP'ILLEGAL NO. OF SORT DISCS'
      READ(1,1000) NDISK,(IDISK(I),I=1,NDISK)
      READ(1,1000) NPROC
      IF(NPROC.EQ.0) GOTO 340
      IF(NPROC.LT.0.OR.NPROC.GT.4) STOP'ILLEGAL NO. OF TEXT LINES'
      BACKSPACE 1
      READ(1,1200) NPROC,((PROC(I,J),I=1,64),J=1,NPROC)
 1200 FORMAT(I6,(64A1))
 340  CLOSE(UNIT=1)
C
C  SET POSITION VECTOR FOR OUTPUT TRACES
C
      DO 105 I=1,NTRACO
      DO 105 J=1,NSPLIT
 105  IPOS((I*NSPLIT)-J+1)=NROW-IFILE((I-1)*NSPLIT+J)+1
C
C  SET UP VELOCITY ANALYSIS CDP NUMBERS IF NECESSARY
C
      IF(VELNUM.GE.0) GO TO 100
      VELAN(1)=NVSRT
      DO 110 I=2,100
 110  VELAN(I)=VELAN(I-1)+NVINT
C
C  SET UP TRACE EDIT REQUEST 1=EDIT 0=NO EDIT
C
 100  DO 120 I=1,48
 120  IEDTR(I)=0
      IF(NTRED.LE.0) GO TO 140
      DO 130 I=1,NTRED
 130  IEDTR(NEDTR(I))=1
C
C  SET UP OUTPUT TRACE REQUEST 1=REQUIRED 0=NOT REQUIRED
C
 140  DO 150 I=1,NTRACI
 150  ITRO(I)=0
      DO 160 I=1,NTRACO*NSPLIT
 160  ITRO(ITRACO(I))=1
C
C  SET UP CONSTANTS
C
      IF(NSMPIN.GT.4096) STOP'INPUT TRACES TOO LONG'
      IWORDR=NSMPIN*2+120
      ISMPIN=(ITFIN-ITSTRT)*FSAMP
      IF(ISMPIN.GT.4096) STOP'TRACE WINDOW TOO LONG'
      ISMPOT=IFIX(ISMPIN/NDEC)
      IWORDW=ISMPOT*2+120
      IBLK=IFIX(IWORDW/256)+1
      DO 155 I=1,NSPLIT
 155  IBLKOF(I)=0
      NTRANF=2
 162  IF(NTRANF.GE.ISMPIN) GO TO 161
      NTRANF=NTRANF*2
      GO TO 162
 161  NTRANF=NTRANF*2
      IF(NDEC.LE.0) NDEC=1
      NTAP=IFIX(NTRANF*0.25/NDEC)
      N3=2*(NTAP+1)
      N4=N3+(2*NTAP)
      NREST=NTRANF+2-N4
      ITIMOT=IFIX(1000.0*NDEC/FSAMP)
      VNUM=1
      VFIL=0
      IEOTR=1
      IEOTW=1
      IEOTV=1
C
C  READ IN HEADER INFO
C
      TYPE 1500
 1500 FORMAT('$GIVE CURRENT OUTPUT REEL NO:')
      ACCEPT 1000,REELNU
```

```
      IF(REELNU.EQ.0) REELNU=1
      TYPE 1600
 1600 FORMAT('$GIVE NEXT OUTPUT TRACE SEQUENCE NUMBER:')
      ACCEPT 1700,FTRCNO
 1700 FORMAT(F15.0)
      IF(FTRCNO.EQ.0.0) FTRCNO=1.0
      IF(NFORM.EQ.4) CALL WESTRT
C
C  GET NEXT SEQUENCE FILE NUMBER FOR VELOCITY FILES IF REQUIRED
C
      IF(VELNUM.EQ.0) GO TO 12
      TYPE 1820
 1820 FORMAT('$GIVE CURRENT VELOCITY OUTPUT REEL NO:')
      ACCEPT 1000,VFLAG
      IF(VFLAG.EQ.0) VFLAG=1
C
C  CHECK FOR SINGLE TRACE I/P - O/P
C
 12   IF(NTRACI.EQ.1.AND.NTRACO.EQ.1) GOTO 50
      IF(NTRACO.EQ.1) GOTO 9
      DO 170 I=2,NTRACO
      DO 170 J=1,NSPLIT
 170  IBLKOF((I-1)*NSPLIT+J)=IBLKOF((I-2)*NSPLIT+J)+IBLK
 9    IBLKW=NTRACO*IBLK
      LSTBLK=IBLKW-1
C
C  SET UP SORT FILE SPECIFICATIONS
C
      ID=1
      CALL SCOPY('DR SRTFIL   ',SRTRT,12)
      DO 10 I=1,NROW
      L=I
      ENCODE(3,1001,SRTRT(10),ERR=11)L
 1001 FORMAT(I3)
 17   IF(SRTRT(10).NE."40) GO TO 14
      DO 16 L=10,12
 16   SRTRT(L)=SRTRT(L+1)
      GO TO 17
 11   STOP'ENCODE ERROR'
C
C  CHECK AND INSERT DISK ID. NUMBER
C
 14   IF(IDISK(ID).GT.7) STOP'SORT FILES - NO ROOM OR NOT FOUND'
      ENCODE(1,1002,NUM,ERR=11) IDISK(ID)
 1002 FORMAT(I1)
      CALL INSERT(NUM,SRTRT,3,1)
      ITCH=15+I
      CALL IRAD50(12,SRTRT,FNAME)
C
C  CHECK IF RESTART OF PROGRAM
C
      IF(USEFLG.NE.0) GO TO 20
C
C  ENTER DISK FILES AT START OF RUN
C
      IER=IENTER(ITCH,FNAME,IBLKW)
      IF(IER.GT.0) GO TO 13
      IF(IER.EQ.-2) GO TO 15
      STOP 'ENTER ERROR'
 13   IF(IWRITW(256,IDUM,LSTBLK,ITCH).LT.0) STOP 'DUMMY WRT ERR'
      CALL CLOSEC(ITCH)
      GO TO 20
C
C  UPDATE DISK ID NUMBER
C
 15   ID=ID+1
      GO TO 14
C
C  LOOKUP DISK FILE
C
 20   IER=LOOKUP(ITCH,FNAME)
      IF(IER.GT.0) GO TO 10
```

```
      IF(IER.EQ.-2) GO TO 15
      STOP 'LOOKUP ERROR'
 10   CONTINUE
C
C  LOOKUP READ TAPE DRIVE
C
 50   ICHN=TPDRR+10
      ILK=LOOKUP(ICHN,MTNAM(TPDRR+1),-1)
      IER=IPEEK("172520)
      IF((IER.AND."100).GT.0) GOTO 60
      TYPE 1850,TPDRR
 1850 FORMAT(' GIVEN READ DRIVE NO.:'I1' NOT AVAILABLE'/
     $'$GIVE NEW READ DRIVE NO.:')
      ACCEPT 1001,TPDRR
      IF(TPDRR.GE.3) STOP'NON-EXISTENT READ DRIVE'
      GOTO 50
C
C  IF LOOKUP OK, CHECK FORMAT AND BOT FLAG
C
 60   IEOTR=0
      IF(NFORM.EQ.3) STOP'NO SEG-D INPUT YET'
      IF(NFORM.EQ.2) STOP'NO LAMMONT INPUT YET'
      IF((IER.AND."40).LE.0) GOTO 80
C
C  READ REEL HEADERS FROM TAPE
C
      IF(IRMTLM(RLHDBL(1,1),ICHN,1600,ITMERR(1)).NE.1600) STOP'ERROR
     + READING ASCII TAPE HEADER'
      IF(NFORM.EQ.1) CALL ISPFNW("376,ICHN)
      IF(IRMTLM(RLHDCD(1),ICHN,200,ITMERR(1)).NE.200) STOP'ERROR
     + READING BINARY TAPE HEADER'
      IF(NFORM.EQ.1) CALL ISPFNW("376,ICHN)
      IF(USEFLG.NE.0) GOTO 80
      IF(NFORM.EQ.0.OR.NFORM.EQ.4) CALL EBCIDC(OUTBUF(1),1600)
      IF(NFORM.EQ.0.OR.NFORM.EQ.4) CALL ARYSWP(OUTBUF(3201),200)
C
C  OPEN DISK FILE FOR PERMANENT STORAGE OF TAPE HEADER BLOCKS
C
      OPEN(UNIT=3,NAME='DK6:SYTPHD.DAT',TYPE='UNKNOWN',ERR=70)
      GO TO 75
 70   STOP'OPEN SYTPHD ERROR'
 75   WRITE (3,2000)((RLHDBL(I,J),I=1,80),J=1,40)
 2000 FORMAT(' ',80A1)
      WRITE(3,2100)(RLHDCD(I),I=1,200)
 2100 FORMAT(' ',8O6)
      CLOSE(UNIT=3)
 80   CONTINUE
C
C  OPEN ERROR LOG FILE
C
      OPEN(UNIT=2,NAME='DK6:SORT.LOG',TYPE='NEW',ERR=220)
      GO TO 230
 220  STOP'LOG OPEN ERROR'
C
C  WRITE TITLE TO LOG FILE
C
 230  WRITE(2,2200)
 2200 FORMAT(' SORT LOG FILE',/,
     +8X,'1',4X,'2',4X,'3',4X,'4',4X,'5',4X,'6',4X,'7',4X,'8',
     $4X,'9',3X,'10',3X,'11',3X,'12',3X,'13',3X,'14',3X,'15',3X,'16',
     $3X,'17',3X,'18',3X,'19',3X,'20',3X,'21',3X,'22',3X,'23',3X,'24',
     $3X,'25',3X,'26',3X,'27',3X,'28',3X,'29',3X,'30',3X,'31',3X,'32',
     $3X,'33',3X,'34',3X,'35',3X,'36',3X,'37',3X,'38',3X,'39',3X,'40',
     $3X,'41',3X,'42',3X,'43',3X,'44',3X,'45',3X,'46',3X,'47',3X,'48')
      RETURN
      END
```

```
      SUBROUTINE SORTHD(ICDPNO,ITIMOT,REELNU,TPDR,IEOT,ISTOP)
C
C TPDR WRITE DRIVE NUMBER
C
C ICDPNO CURRENT FILE NUMBER
C
C IEOT END OF TAPE FLAG
C
      REAL*8 MTNAM(3)
      LOGICAL*1 RLHDBL(80,40),CLIENT(24),LINE(12),AREA(23),
     %REELNO(11),DAY(4),YEAR(5),OBSVER(17),TRINRC(8),CDPFLD(10),
     %SMPINT(9),SPINTR(7),BITS(6),BYTES(7),RECFMT(8),RLFMT(8),
     %UNITS(10),SFLTPT(5),GFLTPT(5),SOURCE(12),SORCNO(8),SCPTIN(18),
     %SWIDTH(18),SRCPTN(27),SPDOFS(8),SPDDIS(8),GRPINT(20),GEOGPN(5),
     %PROC1(64),PROC2(64),PROC3(76),PROC4(76),PROC5(76),AREANO(24),
     %DAYNO(5),YEARNO(6),OBSVNO(18),RLHDCD(400),OUTBUF(4156),
     +PROC(64,4),TRCREV(33)
      INTEGER*2 LINENO,REELNU,NCHAN,NSAMP,IGUNNO,SLSPAC,IWIDTH,
     %SROFF,ITOTAL,TRCNUM,AUXTRS,SAMPIN,ORSINT,NOSMPL,ORSAMP,
     %FMTCD,CDPNO,SORTCD,VERTSM,BINGAN,AMPREC,UNITNO,POLARY,
     +IBLKOF(48),TPDR,TPDRW,ITMERR(2)
      INTEGER*4 LNNOL,RLNOL
      COMMON /DATBUF/OUTBUF
      COMMON /REELS/PROC,NPROC,NREV,NBLKTY
      COMMON /OUTPUT/NTRACO,IBLKOF,IWORDW,FTRCNO,TPDRW,IEOTW,ISMPOT,NROW
      COMMON /TAPE/ MTNAM,ITMERR
      EQUIVALENCE(OUTBUF(1),RLHDBL(1,1)),(OUTBUF(3201),RLHDCD(1))
      EQUIVALENCE (RLHDBL(11,1),CLIENT(1)),(RLHDBL(9,2),LINE(1)),
     %(RLHDBL(26,2),AREA(1)),(RLHDBL(12,3),REELNO(1)),
     %(RLHDBL(41,3),DAY(1)),(RLHDBL(50,3),YEAR(1)),
     %(RLHDBL(64,3),OBSVER(1)),(RLHDBL(23,5),TRINRC(1)),
     %(RLHDBL(71,5),CDPFLD(1)),(RLHDBL(20,6),SMPINT(1)),
     %(RLHDBL(42,6),SPINTR(1)),(RLHDBL(56,6),BITS(1)),
     %(RLHDBL(74,6),BYTES(1)),(RLHDBL(21,7),RECFMT(1)),
     %(RLHDBL(45,7),RLFMT(1)),(RLHDBL(71,7),UNITS(1)),
     %(RLHDBL(29,8),SFLTPT(1)),(RLHDBL(52,9),GFLTPT(1)),
     %(RLHDBL(17,11),SOURCE(1)),(RLHDBL(41,11),SORCNO(1)),
     %(RLHDBL(63,11),SCPTIN(1)),(RLHDBL(63,12),SWIDTH(1)),
     $(RLHDBL(17,12),SRCPTN(1)),(RLHDBL(19,15),SPDOFS(1)),
     $(RLHDBL(39,15),SPDDIS(1)),(RLHDBL(61,15),GRPINT(1)),
     $(RLHDBL(25,16),GEOGPN(1)),(RLHDBL(17,21),PROC1(1)),
     $(RLHDBL(17,22),PROC2(1)),(RLHDBL(5,23),PROC3(1)),
     $(RLHDBL(5,24),PROC4(1)),(RLHDBL(5,25),PROC5(1))
      EQUIVALENCE (RLHDCD(5),LNNOL),(RLHDCD(9),RLNOL),
     %(RLHDCD(13),TRCNUM),(RLHDCD(15),AUXTRS),(RLHDCD(17),SAMPIN),
     %(RLHDCD(19),ORSINT),(RLHDCD(21),NOSMPL),(RLHDCD(23),ORSAMP),
     %(RLHDCD(25),FMTCD),(RLHDCD(27),CDPNO),(RLHDCD(29),SORTCD),
     %(RLHDCD(31),VERTSM),(RLHDCD(51),BINGAN),(RLHDCD(53),AMPREC),
     %(RLHDCD(55),UNITNO),(RLHDCD(57),POLARY)
      DATA TRCREV/'O','R','I','G','I','N','A','L',' ','T','R',
     +'A','C','E',' ','O','R','D','E','R','I','N','G',' ','R',
     +'E','V','E','R','S','E','D','.'/
C
C     ASSIGN CHANNEL NUMBERS
C
 210  ICHN=TPDR+10
C
C     LOOKUP TAPE DRIVE
C
      TYPE 2000,TPDR,ICDPNO
      WRITE(2,2000) TPDR,ICDPNO
 2000 FORMAT(' START NEW WRITE DRIVE:',I1,' CDP NO.:'I5,//)
      IEOT=0
      ILK=LOOKUP(ICHN,MTNAM(TPDR+1),-1)
      IER=IPEEK("172520)
      ISTUS=IER.AND."100
      IF(ISTUS.GT.0) GOTO 240
      TYPE 2100,TPDR,ICDPNO
 2100 FORMAT(' DRIVE NO:'I1,' N/A AT CDP NO.:',I5)
      TYPE 2200
 2200 FORMAT('$GIVE NEW WRITE DRIVE NO.:')
      ACCEPT 2300,TPDR
```

```
 2300 FORMAT(I1)
      IF(TPDR.LE.2) GO TO 210
      ISTOP=1
      RETURN
C
C     CHECK WRITE DRIVE IS NOT WRITE PROTECTED
C
 240  ISTUS=IER.AND."4
      IF(ISTUS.EQ.0) GOTO 220
      TYPE 2900,TPDR
 2900 FORMAT(' WRITE DRIVE NO:',I1,' IS WRT PROT')
      PAUSE
      GO TO 210
 220  ISTUS=IER.AND."40
      IF(ISTUS.EQ.0) RETURN
C
C  READ TAPE HEADER BLOCKS FROM DISK
C
      OPEN(UNIT=3,NAME='DK6:SYTPHD.DAT',TYPE='OLD',ERR=100)
      GO TO 110
 100  TYPE 1010
 1010 FORMAT(' NO TAPE HEADERS ON DISK')
      DO 105 I=1,3600
 105  OUTBUF(I)=0
      GO TO 120
 110  READ(3,1020)((RLHDBL(I,J),I=1,80),J=1,40)
 1020 FORMAT(' ',80A1)
      READ(3,1025)(RLHDCD(I),I=1,200)
 1025 FORMAT(' ',806)
      CLOSE(UNIT=3)
C
C  UPDATE REEL HEADERS
C
 120  ENCODE(11,1030,REELNO,ERR=30) REELNU
      ENCODE(8,1035,TRINRC,ERR=30) NTRACO
      ENCODE(9,1030,SMPINT,ERR=30) ITIMOT
      ENCODE(7,1035,SPINTR,ERR=30) ISMPOT
      ENCODE(10,1030,CDPFLD,ERR=30) NTRACO
 1030 FORMAT(I7)
 1035 FORMAT(I6)
      DO 40 I=1,NPROC
      DO 40 J=1,64
 40   RLHDBL(15+J,24+I)=PROC(J,I)
      IF(NREV.NE.1) GOTO 50
      DO 45 I=1,33
 45   RLHDBL(15+I,24+NPROC+1)=TRCREV(I)
      GO TO 50
 30   ISTOP=2
      RETURN
 50   IF(NBLKTY.EQ.1) WRITE(5,1020) ((RLHDBL(I,J),I=1,80),J=1,40)
      NBLKTY=0
      IF(IWMTLM(RLHDBL(1,1),ICHN,1600,ITMERR(1)).NE.1600) STOP'ERROR
     + WRITING ASCII HEADER'
      CALL ISPFNW("377,ICHN)
      CALL JICVT(REELNU,RLNOL)
      CALL JJCVT(RLNOL)
      TRCNUM=NTRACO
      SAMPIN=ITIMOT
      NOSMPL=ISMPOT
      CDPNO=NTRACO
      SORTCD=2
      IF(IWMTLM(RLHDCD(1),ICHN,200,ITMERR(1)).NE.200) STOP'ERROR
     + WRITING BINARY TAPE HEADER'
      CALL ISPFNW("377,ICHN)
      RETURN
      END
```

```
      SUBROUTINE SRTSUB(ICDPNO,REELNU,ISTOP)
C
      VIRTUAL SEIS(4156)
      INTEGER*2 INDEX(60),IPOS(48),IBLKOF(48),TPDRR,TPDRW,
     +ITRO(48),IEDTR(48),ISUM(48),REELNU,VELAN(101),VNUM,
     +SSMPOT,USEFLG,TPDRV,VFIL,VFLAG,VELNUM,ITMERR(2)
      INTEGER*4 JFNUM,JTRAS,JCDPNO
      REAL*8 MTNAM(3)
      REAL*4 SEIS,TPHDBF(60),SWPBUF(1039)
      LOGICAL*1 OUTBUF(4156),DEL(48),PROC(64,4),SPACE,STAR,IANS
      COMMON /DATBUF/OUTBUF
      COMMON /OUTPUT/NTRACO,IBLKOF,IWORDW,FTRCNO,TPDRW,IEOTW,ISMPOT,NROW
      COMMON /INPUT/IWORDR,TPDRR,IEOTR,NFILES,NTRACI,NSMPIN,NFORM,
     +INDEX,IPOS,VELAN,VNUM,ITSTRT,USEFLG,ISMPIN,NSPLIT
      COMMON/WEST/AWESTN
      COMMON /PROCES/NDEC,NSPK,THRESH,IEDTR,ITRO,FSAMP,ITIMOT,
     +NTAP,NTRANF,N3,N4,NREST
      COMMON /REELS/PROC,NPROC,NREV,NBLKTY
      COMMON /VELOCY/TPDRV,IEOTV,VFIL,VELNUM,VFLAG
      COMMON /TAPE/MTNAM,ITMERR
      EQUIVALENCE (OUTBUF(1),TPHDBF(1)),(OUTBUF(9),JFNUM),
     +(OUTBUF(13),JTRAS),(OUTBUF(21),JCDPNO),(OUTBUF(29),ITYPE),
     +(OUTBUF(109),IDELAY),(OUTBUF(115),ISAMP),(OUTBUF(117),ITIME),
     +(OUTBUF(1),SWPBUF(1))
      DATA STAR/'*'/,SPACE/' '/
      ICHN=10+TPDRR
      SSMPOT=IFIX(ITSTRT*FSAMP)
C
C START OF MAIN WORK LOOP
C
      DO 999 IFIL=1,NFILES
      IFILL=IFIL
      DO 120 I=1,NSPLIT
      INDEX(NROW+1)=INDEX(1)
      DO 100 J=1,NROW
  100 INDEX(J)=INDEX(J+1)
  120 CONTINUE
      ITRIN=0
      IF(NREV.EQ.1) ITRIN=NTRACI+1
C
C MAIN TRANSFER LOOP
C
      DO 200 ITRS=1,NTRACI
      INTR=ITRS
      DEL(INTR)=SPACE
      ISUM(INTR)=0
      IANS='N'
C
C  READ DATA TRACE FROM READ DRIVE TO MEMORY
C
  240 IRD=IRMTXM(SEIS(1),ICHN,IWORDR,ITMERR(1))
      IF(IRD.EQ.IWORDR) GO TO 210
      IF(IRD.NE.-1) GO TO 235
      CALL ISPFN ("372,ICHN)
      TYPE 2100,TPDRR
 2100 FORMAT(' END OF READ TAPE ON DRIVE:',I1,)
  230 CALL CLOSEC(ICHN)
      TYPE 2105
 2105 FORMAT('$GIVE NEW READ DRIVE NO:')
      ACCEPT 1100,TPDRR
 1100 FORMAT(I1)
      IF(TPDRR.LE.2) GOTO 225
      ISTOP=12
      RETURN
  225 ICHN=10+TPDRR
      ILK=LOOKUP(ICHN,MTNAM(TPDRR+1),-1)
      IER=IPEEK("172520)
      IF((IER.AND."100).GT.0) GO TO 220
      TYPE 2110,TPDRR
 2110 FORMAT(' READ DRIVE NO:',I1,' N/A')
      GO TO 230
C
```

```
C  READ HEADER BLOCKS FROM NEW READ DRIVE
C
 220  TYPE 2120,TPDRR,ICDPNO
      WRITE(2,2120) TPDRR,ICDPNO
 2120 FORMAT(' START NEW READ DRIVE:',I1,' CDP NUMBER:',I5,//)
      IF(IRMTLM(OUTBUF(1),ICHN,1600,ITMERR(1)).NE.1600) GO TO 236
      IF(NFORM.EQ.1) CALL ISPFNW("376,ICHN)
      IF(IRMTLM(OUTBUF(1),ICHN,200,ITMERR(1)).NE.200) GO TO 236
      IF(NFORM.EQ.1) CALL ISPFNW("376,ICHN)
      GO TO 240
 235  TYPE 2130,TPDRR
 2130 FORMAT(' ERROR ON READ DRIVE:',I1,/,'$CONTINUE(Y/N)?')
      ACCEPT 1110,IANS
 1110 FORMAT(A1)
      IF(IANS.EQ.'Y') GOTO 210
      ICDPNO=ICDPNO-INTR+1
      ISTOP=11
      RETURN
 236  ISTOP=13
      RETURN
C
C  SKIP INTER-TRACE TAPE MARK ON DURHAM SEG-Y
C
 210  IF(NFORM.EQ.1)CALL ISPFNW("376,ICHN)
C
C  IDENTIFY INPUT TRACE
C
 260  IF(NFORM.NE.0.AND.NFORM.NE.4) GOTO 263
      DO 273 I=4,1,-1
      J=(I-1)*1039
      DO 272 K=1,1039
 272  SWPBUF(K)=SEIS(J+K)
      CALL ARYSWP(SWPBUF(1),2078)
      DO 273 K=1,1039
 273  SEIS(J+K)=SWPBUF(K)
      GOTO 265
 263  DO 261 LL=1,60
 261  TPHDBF(LL)=SEIS(LL)
 265  IF(IANS.EQ.'Y') ITYPE=2
      CALL JJCVT(JFNUM)
      IF(NFORM.EQ.4) CALL WESTEN
      CALL IJCVT(JFNUM,IFNUM)
      CALL JJCVT(JTRAS)
      CALL IJCVT(JTRAS,ITRAS)
      IF(ITRAS.LE.0.OR.ITRAS.GT.NTRACI) ITRAS=1
C
C  CHECK IF INPUT TRACE REQUIRED FOR OUTPUT
C
      IF(ITRO(ITRAS).EQ.0) GO TO 200
      IF(NREV.NE.1)ITRIN=ITRIN+1
      IF(NREV.EQ.1) ITRIN=ITRIN-1
      IF(NROW.EQ.1.AND.NTRACO.GT.1) GOTO 262
      ICDPNO=(NSPLIT*IFNUM)-NROW+IPOS(ITRIN)
      GOTO 264
 262  CALL JJCVT(JCDPNO)
      CALL IJCVT(JCDPNO,ICDPNO)
 264  IF(ICDPNO.LE.0.AND.USEFLG.EQ.0.AND.NTRACI.GT.1) GO TO 200
C
C  CHECK 'ITYPE' FOR DUD TRACES OR EDIT REQUESTS
C
      IF(ITYPE.GE.2) GO TO 270
      IF(IEDTR(ITRAS).NE.0) GO TO 275
C
C   DO PROCESSING IN AP FOR DECIMATION AND GLITCH REMOVAL
C
      CALL VCLR(0,1,4096)
      JSMPOT=1
      ISMPDL=IFIX(IDELAY*FSAMP)
      IAPST=ISMPDL-SSMPOT
      IF(IAPST.LT.0) JSMPOT=1-IAPST
      IF(IAPST.LT.0) IAPST=0
      ISMP=MIN0(SSMPOT+ISMPIN,ISMPDL+NSMPIN)-MAX0(SSMPOT,ISMPDL)
```

```
      CALL APWR
      IF(NFORM.EQ.1)CALL APPTXM(SEIS(JSMPOT+60),IAPST,ISMP,2)
      IF(NFORM.NE.1)CALL APPTXM(SEIS(JSMPOT+60),IAPST,ISMP,3)
      CALL APWD
      IF(NSPK.LT.0.AND.NDEC.EQ.1) GOTO 280
      CALL PRCESS(ISMPIN,IFLAG,ISUM(INTR))
      IF(IFLAG.EQ.0) GO TO 280
C
C  IF TRACE HAS TOO MANY ERRORS DELETE AND SET ITYPE IN TRACE HEADER
C
 275  ITYPE=2
 270  DEL(INTR)=STAR
      CALL VCLR(0,1,ISMPOT)
      CALL APWR
C
C  RETRIEVE DATA FROM AP
C
 280  CALL APGTXM(SEIS(61),0,ISMPOT,2)
      CALL APWD
C
C  UPDATE TRACE HEADER
C
 205  ITIME=ITIMOT
      IDELAY=ITSTRT
      ISAMP=ISMPOT
      CALL JICVT(ICDPNO,JCDPNO)
      CALL JICVT(ITRIN,JCDPTR)
      CALL JJCVT(JCDPNO)
      CALL JJCVT(JCDPTR)
      CALL JJCVT(JFNUM)
      CALL JJCVT(JTRAS)
      DO 206 LL=1,60
 206  SEIS(LL)=TPHDBF(LL)
C
C  CHECK FOR SINGLE TRACE I/P - O/P
C
      IF(NTRACI.EQ.1.AND.NTRACO.EQ.1) GOTO 200
C
C   WRITE OUTPUT DATA TO DISK SORT MATRIX FILE
C
      ITCH=15+INDEX(IPOS(ITRIN))
      ITBLKO=IBLKOF(ITRIN)
C      TYPE *,IFNUM,ITRAS,ICDPNO,ITRIN,IPOS(ITRIN),INDEX(IPOS(ITRIN)),
C     +ITCH,ITBLKO
      IF(IWDRXM(SEIS(1),ITCH,IWORDW,ITBLKO,0).EQ.IWORDW) GO TO 200
      ISTOP=14
      RETURN
  200 CONTINUE
C
C  CORRECT CURRENT CDP NO. IN TRACE REVERSE CASE
C
      IF(NREV.EQ.1) ICDPNO=ICDPNO-NROW+1
C
C  WRITE OUT LOG FILE DATA
C
      IF(NSPK.GE.0) WRITE(2,2200)IFNUM,((ISUM(I),DEL(I)),I=1,NTRACI)
 2200 FORMAT( 1X,I4,48(I4,A1))
C
C  CHECK IF FULL CDP GATHER READY FOR OUTPUT
C
C      TYPE *,USEFLG,IFILL,NROW
      IF(USEFLG.EQ.0
     $          .AND.IFILL*NSPLIT.LT.NROW)GOTO 990
C
C  OUTPUT
C
      DO 989 I=1,NSPLIT
      ITCH=INDEX(I)+15
C
C  TRANSFER CDP GATHER FROM DISK TO TAPE
C
 960  CALL SRTOUT(SEIS,ICDPNO,NTRACI,NSPLIT,ITCH,ISTOP)
```

```
      IF(ISTOP.NE.0) RETURN
      IF(IEOTW.EQ.0) GO TO 950
      REELNU=REELNU+1
      CALL SORTHD(ICDPNO,ITIMOT,REELNU,TPDRW,IEOTW,ISTOP)
      IF(ISTOP.NE.0) RETURN
      GO TO 960
C
C  CHECK IF VELOCITY ANALYSIS REQUIRED AT THIS CDP GATHER
C
 950  IF(VELAN(VNUM).EQ.0) GO TO 989
      IF(ICDPNO.LT.VELAN(VNUM)) GO TO 989
C
C  UPDATE VELOCITY ANALYSIS FILE NUMBERS WHEN RESTARTING PROGRAM
C
      IF(ICDPNO.EQ.VELAN(VNUM)) GO TO 985
      VNUM=VNUM+1
      GO TO 950
C
C  WRITE GATHER FILE FOR VELOCITY ANALYSIS IN SEG-Y FORMAT TO TAPE ONLY
C
 985  CALL SRTVSY(SEIS,ICDPNO,NTRACI,NSPLIT,ITCH,ISTOP)
      IF(ISTOP.NE.0) RETURN
      IF(IEOTV.EQ.0) GO TO 989
      VFLAG=VFLAG+1
      CALL SORTHD(ICDPNO,ITIMOT,VFLAG,TPDRV,IEOTV,ISTOP)
      IF(ISTOP.NE.0) RETURN
      GO TO 985
 989  CONTINUE
C
C   END OF MAIN LOOP
C
 990  IF(MOD(IFILL,10).EQ.0) TYPE 9000,IFILL,ICDPNO
 9000 FORMAT('+NO. OF I/P FILES SORTED:',I5,' CURRENT CDP NO.:',I5)
 999  CONTINUE
      IF(USEFLG.EQ.0
     $          .AND.IFILL*NSPLIT.LT.NROW) ISTOP=15
      RETURN
      END




      SUBROUTINE SRTOUT(SEIS,ICDPNO,NTRACI,NSPLIT,ITCH,ISTOP)
C
C NTRACO NUMBER OF OUTPUT TRACES PER FILE
C
C IBLKOF POSITIONS OF OUTPUT TRACES IN DISC FILE
C
C OUTBUF DATA TRANSFER ARRAY
C
C TPDRW WRITE DRIVE NUMBER
C
C ITCH IS THE OUTPUT TRACE FOR DISC READS
C
C FTRCNO CURRENT TRACE NUMBER
C
C IWORDW NUMBER OF DATA WORDS PER TRACE
C
C ICDPNO CURRENT CDP GATHER NUMBER
C
C IEOTW END OF TAPE FLAG
C
      VIRTUAL SEIS(4156)
      REAL*8 MTNAM(3)
      REAL*4 SEIS,UPDATE
      INTEGER*4 JTRCNO,JCDPNO,JCDPTR
      INTEGER*2 IBLKOF(48),TPDRW,ITMERR(2)
      LOGICAL*1 OUTBUF(4156)
      COMMON /DATBUF/ OUTBUF
      COMMON /OUTPUT/NTRACO,IBLKOF,IWORDW,FTRCNO,TPDRW,IEOTW,ISMPOT,NROW
      COMMON /TAPE/ MTNAM,ITMERR
```

```
      EQUIVALENCE (JTRCNO,OUTBUF(1)),(OUTBUF(1),UPDATE),
     +(JCDPNO,OUTBUF(21)),(OUTBUF(21),CDPNO),
     +(JCDPTR,OUTBUF(25)),(OUTBUF(25),CDPTR)
C       ITRY=0
C
C      ASSIGN CHANNEL NUMBERS
C
       ICHN=TPDRW+10
C
C TRANFER DATA IN BLOCKS OF ONE CHANNEL PLUS H/BLK
C
       DO 200 I=1,NTRACO
       IITRC=I
C
C  CHECK FOR SINGLE TRACE I/P - O/P
C
       IF(NTRACI.EQ.1.AND.NTRACO.EQ.1) GOTO 205
C
C  READ TRACE FROM DISC
C
       IBLK=IBLKOF((IITRC-1)*NSPLIT+1)
       IF(IRDRXM(SEIS(1),ITCH,IWORDW,IBLK,0).EQ.IWORDW) GO TO 205
       ISTOP=21
       RETURN
C
C  WRITE TRACE SEQUENCE NUMBER IN HEADER BLOCK
C
 205   CALL JAFIX(FTRCNO,JTRCNO)
       CALL JJCVT(JTRCNO)
       SEIS(1)=UPDATE
       CALL JICVT(IITRC,JCDPTR)
       CALL JJCVT(JCDPTR)
       SEIS(7)=CDPTR
       CDPNO=SEIS(6)
       CALL JJCVT(JCDPNO)
       CALL IJCVT(JCDPNO,ICDPNO)
C
C  WRITE TRACE TO O/P TAPE
C
 220   IWT=IWMTXM(SEIS(1),ICHN,IWORDW,ITMERR(1))
C
C CHECK FOR ERRORS
C
       IF(IWT.EQ.IWORDW) GOTO 210
       IF(IWT.EQ.-1) GOTO 400
C
C RE-TRY
C
   240 ITRY=ITRY+1
       CALL ISPFNW("375,ICHN)
       CALL ISPFNW("376,ICHN)
       WRITE(2,2200) TPDRW,ICDPNO,IITRC,ITMERR(1)
       TYPE 2200,TPDRW,ICDPNO,IITRC,ITMERR(1)
 2200  FORMAT(' ERROR ON DRIVE:',I2,' CDP NO:',I5,' TRACE NO:',I5,
      $' STATUS:',O6,//)
       IF(ITRY.LT.3) GO TO 220
C
C BAD OUTPUT FILE SO REWIND TO LAST FILE AND CLOSE TAPE
C
       CALL BACKSP(IITRC,ICHN)
       FTRCNO=FTRCNO-FLOAT(IITRC)+1.0
       CALL ISPFNW("377,ICHN)
       ISTOP=22
       RETURN
C
C COME HERE TO WRITE TAPE MARK
C
 210   CALL ISPFNW("377,ICHN)
       ISTUS=IPEEK("172520)
       IF((ISTUS.AND."40000).LE.0) GO TO 240
       IF((ISTUS.AND."2000).EQ.0) GOTO 200
       CALL ISPFNW("375,ICHN)
```

```
      GOTO 400
 200  FTRCNO=FTRCNO+1.0
C
C NORMAL RETURN
C
      RETURN
C
C COME HERE FOR EOT
C
 400  IEOTW=1
      CALL BACKSP(IITRC,ICHN)
      FTRCNO=FTRCNO-FLOAT(IITRC)+1.0
C
C CLOSE REWIND OUTPUT TAPE
C
      CALL ISPFNW("377,ICHN)
      CALL ISPFN ("372,ICHN)
      CALL CLOSEC(ICHN)
      TYPE 2300,TPDRW,(ICDPNO-1)
      WRITE(2,2300) TPDRW,(ICDPNO-1)
 2300 FORMAT(' EOT ON WRITE DRIVE:',I2,' CDP NO:',I5,)
      TYPE 2310
 2310 FORMAT('$GIVE NEW WRITE DRIVE NUMBER:')
      ACCEPT 1000,TPDRW
 1000 FORMAT(I1)
      IF(TPDRW.LE.2) RETURN
      ISTOP=23
      RETURN
      END




      SUBROUTINE SRTVSY(SEIS,ICDPNO,NTRACI,NSPLIT,ITCH,ISTOP)
C
C ITCH IS THE OUTPUT TRACE FOR DISC READS
C     ISTOP IS THE ERROR FLAG RETURN TO THE MAIN PROGRAM
C
      VIRTUAL SEIS(4156)
      REAL*8 MTNAM(3)
      REAL*4 SEIS,UPDATE
      INTEGER*4 JTRCNO,JCDPTR
      INTEGER*2 IBLKOF(48),TPDRW,TPDRV,VFIL,VELNUM,VFLAG,ITMERR(2)
      LOGICAL*1 OUTBUF(4156)
      COMMON /DATBUF/ OUTBUF
      COMMON /OUTPUT/NTRACO,IBLKOF,IWORDW,FTRCNO,TPDRW,IEOTW,ISMPOT,NROW
      COMMON /VELOCY/TPDRV,IEOTV,VFIL,VELNUM,VFLAG
      COMMON /TAPE/ MTNAM,ITMERR
      EQUIVALENCE (OUTBUF(1),JTRCNO),(OUTBUF(1),UPDATE),
     +(OUTBUF(25),JCDPTR),(OUTBUF(25),CDPTR)
C
      ITRY=0
C
C     ASSIGN CHANNEL NUMBERS
C
      ICHN=TPDRV+10
C
C TRANFER DATA IN BLOCKS OF ONE CHANNEL PLUS H/BLK
C
      DO 200 I=1,NTRACO
      IITRC=I
C
C  CHECK FOR SINGLE TRACE I/P - O/P
C
      IF(NTRACI.EQ.1.AND.NTRACO.EQ.1) GOTO 205
C
C  READ TRACE FROM DISC
C
      IBLK=IBLKOF((IITRC-1)*NSPLIT+1)
      IF(IRDRXM(SEIS(1),ITCH,IWORDW,IBLK,0).EQ.IWORDW) GO TO 205
      ISTOP=41
```

```
      RETURN
C
C  BLANK TRACE SEQUENCE NUMBER IN VELOCITY HEADER
C
 205  IBLANK=0
      CALL JICVT(IBLANK,JTRCNO)
      SEIS(1)=UPDATE
      CALL JICVT(IITRC,JCDPTR)
      CALL JJCVT(JCDPTR)
      SEIS(7)=CDPTR
 220  IWT=IWMTXM(SEIS(1),ICHN,IWORDW,ITMERR(1))
C
C CHECK FOR ERRORS
C
      IF(IWT.EQ.IWORDW) GOTO 210
      IF(IWT.EQ.-1) GOTO 400
C
C RE-TRY
C
  240 ITRY=ITRY+1
      CALL ISPFNW("375,ICHN)
      CALL ISPFNW("376,ICHN)
      WRITE(2,2200) TPDRV,ICDPNO,I,ITMERR(1)
      TYPE 2200,TPDRV,ICDPNO,I,ITMERR(1)
 2200 FORMAT(' ERROR ON VELOCITY DRIVE:',I2,' VELOCITY FILE CDP NO:',I4,
     +' TRACE NO:',I4,' STATUS:',O6,//)
      IF(ITRY.LT.3) GO TO 220
C
C BAD OUTPUT FILE SO REWIND TO LAST FILE AND CLOSE TAPE
C
      CALL BACKSP(IITRC,ICHN)
      CALL ISPFNW("377,ICHN)
      ISTOP=42
      RETURN
C
C COME HERE TO WRITE TAPE MARK
C
 210  CALL ISPFNW("377,ICHN)
      ISTUS=IPEEK("172520)
      IF((ISTUS.AND."40000).LE.0) GO TO 240
      IF((ISTUS.AND."2000).EQ.0) GOTO 200
      CALL ISPFNW("375,ICHN)
      GOTO 400
 200  CONTINUE
C
C NORMAL RETURN
C
      VFIL=ICDPNO
      RETURN
C
C COME HERE FOR EOT
C
 400  IEOTV=1
      CALL BACKSP(IITRC,ICHN)
C
C CLOSE REWIND OUTPUT TAPE
C
      CALL ISPFNW("377,ICHN)
      CALL ISPFN ("372,ICHN)
      CALL CLOSEC(ICHN)
      TYPE 2300,TPDRV,ICDPNO
      WRITE(2,2300) TPDRV,ICDPNO
 2300 FORMAT(' EOT ON VELOCITY DRIVE:',I2,' VELOCITY FILE CDP
     + NO:',I4,)
      TYPE 2040
 2040 FORMAT('$GIVE NEW VELOCITY WRITE DRIVE NUMBER:')
      ACCEPT 1000,TPDRV
 1000 FORMAT(I1)
      IF(TPDRV.LE.2) RETURN
      ISTOP=43
      RETURN
      END
```

```
      SUBROUTINE SRTEND(ISTOP,ICDPNO)
C
C
C
      LOGICAL*1 SORTHD(6),SRTSUB(6),SRTOUT(6),SRTVSY(6),PROC(64,4)
      INTEGER*2 TPDRR,TPDRW,TPDRV,INDEX(60),IPOS(48),IBLKOF(48),VFIL,
     +VELNUM,VNUM,VELAN(101),USEFLG,VFLAG
      COMMON /OUTPUT/NTRACO,IBLKOF,IWORDW,FTRCNO,TPDRW,IEOTW,ISMPOT,NROW
      COMMON /VELOCY/TPDRV,IEOTV,VFIL,VELNUM,VFLAG
      COMMON /INPUT/IWORDR,TPDRR,IEOTR,NFILES,NTRACI,NSMPIN,NFORM,
     +INDEX,IPOS,VELAN,VNUM,ITSRT,USEFLG,ISMPIN,NSPLIT
      COMMON/WEST/AWESTN
      COMMON /REELS/PROC,NPROC,NREV,NBLKTY
C
C
C
      CALL SCOPY('SORTHD',SORTHD,6)
      CALL SCOPY('SRTSUB',SRTSUB,6)
      CALL SCOPY('SRTOUT',SRTOUT,6)
      CALL SCOPY('SRTVSY',SRTVSY,6)
C
C
C
      IF(ISTOP.EQ.0) GO TO 500
      IF(ISTOP.GT.40) GO TO 100
      DO 45 I=1,NSPLIT
      DO 40 J=NROW,1,-1
 40   INDEX(J+1)=INDEX(J)
 45   INDEX(1)=INDEX(NROW+1)
      IF(ISTOP.GT.20) GO TO 140
      IF(ISTOP.GT.10) GO TO 160
C
C
C
      TYPE 1000,(SORTHD(I),I=1,6),ISTOP
      WRITE(2,1000)(SORTHD(I),I=1,6),ISTOP
 1000 FORMAT(/' TERMINATION FROM SUBROUTINE ',6A1,' ISTOP=',I2)
      GO TO (200,210) ISTOP
 160  TYPE 1000,(SRTSUB(I),I=1,6),ISTOP
      WRITE(2,1000)(SRTSUB(I),I=1,6),ISTOP
      GO TO (220,230,240,250,260)(ISTOP-10)
 140  TYPE 1000,(SRTOUT(I),I=1,6),ISTOP
      WRITE(2,1000)(SRTOUT(I),I=1,6),ISTOP
      GO TO (265,270,200)(ISTOP-20)
 100  TYPE 1000,(SRTVSY(I),I=1,6),ISTOP
      WRITE(2,1000)(SRTVSY(I),I=1,6),ISTOP
      GO TO (330,270,400)(ISTOP-40)
 200  ICDPNO=ICDPNO-1
      WRITE(2,2000)
      TYPE 2000
 2000 FORMAT(' WRITE DRIVE EXIT - REWIND READ TAPE 1 FILE')
      GO TO 510
 210  WRITE(2,2010)
      TYPE 2010
 2010 FORMAT(' TAPE HEADER ENCODE ERROR - REWIND READ TAPE 1 FILE')
      GO TO 510
 220  WRITE(2,2020)
      TYPE 2020
 2020 FORMAT(' TRACE READ ERROR - REWIND TAPE TO BEGINNING OF FILE')
      GO TO 510
 230  WRITE(2,2030)
      TYPE 2030
 2030 FORMAT(' READ DRIVE EXIT')
      GO TO 510
 240  WRITE(2,2040)
      TYPE 2040
 2040 FORMAT(' REEL HEADER READ ERROR')
      GO TO 510
 250  WRITE(2,2050)
      TYPE 2050
 2050 FORMAT(' TRACE DISK WRITE ERROR -',
     +' REWIND READ TAPE TO BEGINNING OF FILE')
```

```
      GO TO 510
 260  WRITE(2,2060)
      TYPE 2060
 2060 FORMAT(' NO OUTPUT FILES WRITTEN. REWIND TAPES AND START AGAIN')
      GO TO 520
 265  ICDPNO=ICDPNO-1
      WRITE(2,2065)
      TYPE 2065
 2065 FORMAT(' DISK READ ERROR - REWIND READ TAPE 1 FILE',/,
     +' REWIND WRITE TAPE TO END OF LAST COMPLETE FILE WRITTEN')
      GO TO 510
 270  ICDPNO=ICDPNO-1
      WRITE(2,2070)
      TYPE 2070
 2070 FORMAT(' BAD O/P TAPE - REWIND READ TAPE 1 FILE')
      GO TO 510
 330  WRITE(2,2130)
      TYPE 2130
 2130 FORMAT(' VELOCITY TRACE READ ERROR')
      GO TO 510
 400  WRITE(2,2200)
      TYPE 2200
 2200 FORMAT(' VELOCITY WRITE DRIVE EXIT - CURRENT VELOCITY FILE LOST')
      GO TO 510
 500  WRITE(2,2210)
      TYPE 2210
 2210 FORMAT(' ***** NORMAL TERMINATION *****')
C
C
C
 510  DO 50 I=1,NROW
 50   CALL CLOSEC(15+I)
      WRITE(2,2220)(FTRCNO-1)
 2220 FORMAT(' LAST OUTPUT TRACE SEQUENCE NUMBER IS:',F15.0,/,
     +' LAST INDEX VALUES ARE:')
      WRITE(2,2230)(INDEX(I),I=1,NROW)
 2230 FORMAT(12I5)
      IF(VELNUM.NE.0)WRITE(2,2250)VFIL
 2250 FORMAT(' CDP NO. OF LAST D-SEG-Y VELOCITY FILE WRITTEN:',I5)
      WRITE(2,2260) ICDPNO
 2260 FORMAT(' CDP NUMBER OF LAST COMPLETE GATHER WRITTEN:',I5)
      IF(NREV.EQ.1) WRITE(2,3000)
 3000 FORMAT(/' ORIGINAL TRACE ORDERING REVERSED DURING THIS SORT')
 520  CLOSE(UNIT=2)
      CALL APRLSE
      RETURN
      END



      SUBROUTINE PRCESS(NSMPIN,IFLAG,ISUM)
C
C DECIMATE DATA IF REQUIRED AND
C CHECK FOR SPIKES IN DATA
C
C  NSMPIN IS THE TOTAL NUMBER OF INPUT DATA SAMPLES
C  NDEC IS THE LEVEL OF DECIMATION
C  THRESH IS THE SPIKE CLIP LEVEL
C  IFLAG IS A FLAG TO THE MAIN PROGRAM THAT A CHANNEL HAS BEEN DELETED
C  NSPK IS THE TOTAL ALLOWABLE NUMBER OF LARGES SPIKES PER CHANNEL
C  NTAP IS THE LENGTH OF COSINE TAPER IN FOR COMPLEX SAMPLES
C  N3 IS THE AP LOCATION THE START OF TAPER,HALF NYQUIST FREQ. O/P
C  N4 IS THE AP LOCATION OF THE END OF THE COSINE TAPER,NYQUIST FREQ. O/P
C  NTRANF IS THE N**2 NUMBER > NSMPIN
C  NREST IS ANY REMAINING DATA ABOVE NYQUIST FREQ. TO BE ZEROED
C
C
      INTEGER*2 IEDTR(48),ITRO(48)
      COMMON /PROCES/NDEC,NSPK,THRESH,IEDTR,ITRO,FSAMP,ITIMOT,
     +NTAP,NTRANF,N3,N4,NREST
```

```
      IFLAG=0
      IF(NSPK.LT.0) GOTO 50
C
C
C DC OFFSET AND SPIKE REMOVAL COMPARISON VALUES
C   TO BE PUT INTO AP
C
      CALL APPUT(THRESH,8192,1,2)
      CALL APWD
      CALL VCLR(8193,1,1)
C
C  CHECK FOR SPIKES OF MAGNITUDE .GT. THRESH
C
      CALL VABS(0,1,4096,1,NSMPIN)
      CALL LVGT(4096,1,8192,0,4096,1,NSMPIN)
C
C  GET TOTAL NUMBER OF SPIKES AND FLAG IN DATA
C
      CALL SVE(4096,1,8193,NSMPIN)
      CALL LVNOT(4096,1,4096,1,NSMPIN)
      CALL VMUL(0,1,4096,1,0,1,NSMPIN)
      CALL VFIX(8193,1,8193,1,1)
      CALL APWR
C
C  GET NUMBER OF ERRORS FROM AP
C
      CALL APGET(ISP,8193,1,1)
      CALL APWD
C
C  CHECK THAT LESS THAN NSPK SPIKES DETECTED, OTHERWISE DELETE
C
      IF(ISP.LE.NSPK) GO TO 50
      CALL VCLR(0,1,NSMPIN)
      CALL APWR
      IFLAG=1
      RETURN
C
C  FFT DATA AND LOWPASS
C
 50   CALL VCLR(NSMPIN,1,8192)
      CALL RFFT(0,NTRANF,1)
      CALL RFFTSC(0,NTRANF,3,1)
      CALL CTAP(N3,N3,0,NTAP)
C
C  CLEAR DC AND ANY DATA ABOVE O/P NYQUIST FREQ.
C
      CALL VCLR(0,1,2)
      CALL VCLR(N4,1,NREST)
      CALL RFFTSC(0,NTRANF,-3,0)
      CALL RFFT(0,NTRANF,-1)
C
C  DO DECIMATION
C
      CALL APWR
      IF(NDEC.EQ.1) RETURN
      NCLR=(NDEC-1)*NSMPIN
      CALL APWAIT
      CALL VCLR(NSMPIN,1,NCLR)
      CALL VMOV(0,NDEC,0,1,NSMPIN)
      CALL APWR
      RETURN
      END
```

```
      SUBROUTINE WESTRT
      REAL*4 AWESTN
      COMMON/WEST/AWESTN
      TYPE 1000
 1000 FORMAT('$GIVE WESTERN''S IDENT NO. OF FIRST FILE IN LINE:')
      ACCEPT 1010,AWESTN
 1010 FORMAT(F15.0)
      AWESTN=(AINT(AWESTN*0.0001))+100.0*AMOD(AWESTN,10000.0)-1.0
      RETURN
      END
```

```
      SUBROUTINE WESTEN
      LOGICAL*1 OUTBUF(4156)
      REAL*4 AWESTN,AFNUM
      INTEGER*4 JFNUM
      COMMON /WEST/AWESTN
      COMMON /DATBUF/OUTBUF
      EQUIVALENCE (OUTBUF(9),JFNUM)
      AFNUM=AJFLT(JFNUM)
      CALL JAFIX((AINT(AFNUM*0.0001))+100.0*AMOD(AFNUM,10000.0)-
     %AWESTN,JFNUM)
      RETURN
      END
```

```
 .TITLE ARYSWP
 .GLOBL ARYSWP
ARYSWP: MOV @4(R5),R4   ;GET WORD COUNT FROM FORTRAN
 MOV 2(R5),R3   ;GET ADDRESS OF FIRST ARRAY
      ;ELEMENT
1$: SWAB (R3)+    ;BYTE-SWAP ARRAY ELEMENT
 DEC R4    ;DECREMENT WORD COUNT
 BGT 1$    ;BRANCH IF MORE WORDS TO DO
 RTS PC
 .END
```

```
 .TITLE EBCIDC
 .GLOBL EBCIDC
EBCIDC: MOV @4(R5),R1  ;GET WORD COUNT
 MOV 2(R5),R3  ;GET ADDRESS OF FIRST ARRAY
     ;ELEMENT
8$: MOV (R3),R0   ;PUT FIRST ARRAY ELEMENT IN R0
 CLR R2
4$: CMPB #100,R0   ;TEST FOR SPACE CHARACTER
 BNE 1$   ;BRANCH IF NOT A SPACE
 BIC #100,R0
 BIS #40,R0
 BR 6$
1$: CMPB #141,R0   ;TEST FOR / CHARACTER
 BNE 9$   ;BRANCH IF NOT /
 BIC #141,R0
 BIS #57,R0
 BR 6$
9$: CMPB #140,R0   ;TEST FOR - CHARACTER
 BNE 10$   ;BRANCH IF NOT -
 BIC #140,R0
 BIS #55,R0
 BR 6$
10$: CMPB #113,R0   ;TEST FOR . CHARACTER
 BNE 7$   ;BRANCH IF NOT .
 BIC #113,R0
 BIS #56,R0
 BR 6$
7$: BIC #200,R0   ;CLEAR BIT 7
 BIT #20,R0   ;TEST BIT 4
 BEQ 2$   ;BRANCH IF BIT 4 NOT SET
 BIT #40,R0   ;TEST BIT 5
 BEQ 3$   ;BRANCH IF BIT 5 NOT SET
 BIC #100,R0   ;CLEAR BIT 6 - CHARACTER IS
     ;NUMERIC
 BR 6$
3$: BIC #20,R0   ;CLEAR BIT 4
 ADD #11,R0   ;ADD 1001
 BR 5$
2$: BIT #40,R0   ;TEST BIT 5
 BEQ 5$   ;BRANCH IF BIT 5 NOT SET
 BIC #40,R0   ;CLEAR BIT 5
 ADD #21,R0   ;ADD 10001
5$: BIT #100,R0   ;TEST BIT 6
 BNE 6$   ;BRANCH IF BIT 6 SET
 BIS #140,R0   ;SET BITS 5 & 6
6$: INC R2
 SWAB R0   ;SWAP BYTES
 CMP #1,R2   ;BOTH BYTES DECODED?
 BEQ 4$   ;BRANCH IF ONLY 1 BYTE DONE
 MOV R0,(R3)+  ;RETURN 2 DECODED BYTES
 DEC R1   ;DECREMENT WORD COUNT
 BGT 8$   ;BRANCH IF MORE WORDS TO DO
 RTS PC
 .END
```

```
      SUBROUTINE PRCESS(NSMPIN,IFLAG,ISUM)
C
C DECIMATE DATA IF REQUIRED AND
C CHECK FOR SPIKES IN DATA
C
C  NSMPIN IS THE TOTAL NUMBER OF INPUT DATA SAMPLES
C  NDEC IS THE LEVEL OF DECIMATION
C  THRESH IS THE SPIKE CLIP LEVEL
C  IFLAG IS A FLAG TO THE MAIN PROGRAM THAT A CHANNEL HAS BEEN DELETED
C  NSPK IS THE TOTAL ALLOWABLE NUMBER OF LARGES SPIKES PER CHANNEL
C  NTAP IS THE LENGTH OF COSINE TAPER IN FOR COMPLEX SAMPLES
C  N3 IS THE AP LOCATION THE START OF TAPER,HALF NYQUIST FREQ. O/P
C  N4 IS THE AP LOCATION OF THE END OF THE COSINE TAPER,NYQUIST FREQ. O/P
C  NTRANF IS THE N**2 NUMBER > NSMPIN
C  NREST IS ANY REMAINING DATA ABOVE NYQUIST FREQ. TO BE ZEROED
C
C
      REAL*4 TLIM(4)
      INTEGER*2 IEDTR(48),ITRO(48)
C
      COMMON /PROCES/NDEC,NSPK,THRESH,IEDTR,ITRO,FSAMP,ITIMOT,
     +NTAP,NTRANF,N3,N4,NREST
C
      IFLAG=0
      IF(NSPK.LT.0) GOTO 50
C
C  **** HEBRIDEAN SPIKE REMOVAL ****
C
      TLIM(1)=0.25
      TLIM(2)=0.2501
      TLIM(3)=0.5
      TLIM(4)=THRESH
      CALL APPUT(TLIM,8192,4,2)
      CALL APWD
      CALL VCLR(8196,1,1)
      CALL RAMP(8192,8194,4097,2,12)
      CALL RAMP(8193,8194,4096,2,12)
      CALL GLITCH(0,1,4096,12,NSMPIN)
      CALL APWR
      CALL APCHK(IGL)
      CALL APWD
C
C  CHECK FOR SPIKES OF MAGNITUDE .GT. THRESH
C
      CALL VABS(0,1,4096,1,NSMPIN)
      CALL LVGT(4096,1,8195,0,4096,1,NSMPIN)
C
C  GET TOTAL NUMBER OF SPIKES AND FLAG IN DATA
C
      CALL SVE(4096,1,8196,NSMPIN)
      CALL LVNOT(4096,1,4096,1,NSMPIN)
      CALL VMUL(0,1,4096,1,0,1,NSMPIN)
      CALL VFIX(8193,1,8193,1,1)
      CALL APWR
C
C  GET NUMBER OF ERRORS FROM AP
C
      CALL APGET(ISP,8196,1,1)
      CALL APWD
C
C  CHECK THAT LESS THAN NSPK SPIKES DETECTED, OTHERWISE DELETE
C
      ISUM=ISP+IGL
      NQUIT=NSMPIN/4
      IF(ISP.LE.NSPK.AND.ISUM.LT.NQUIT) GO TO 50
      CALL VCLR(0,1,NSMPIN)
      CALL APWR
      IFLAG=1
      RETURN
C
C  FFT DATA AND LOWPASS
C
```

```
 50   CALL VCLR(NSMPIN,1,8192)
      CALL RFFT(0,NTRANF,1)
      CALL RFFTSC(0,NTRANF,3,1)
      CALL CTAP(N3,N3,0,NTAP)
C
C  CLEAR DC AND ANY DATA ABOVE O/P NYQUIST FREQ.
C
      CALL VCLR(0,1,2)
      CALL VCLR(N4,1,NREST)
      CALL RFFTSC(0,NTRANF,-3,0)
      CALL RFFT(0,NTRANF,-1)
C
C  DO DECIMATION
C
      CALL APWR
      IF(NDEC.EQ.1) RETURN
      NCLR=(NDEC-1)*NSMPIN
      CALL APWAIT
      CALL VCLR(NSMPIN,1,NCLR)
      CALL VMOV(0,NDEC,0,1,NSMPIN)
      CALL APWR
      RETURN
      END
```

```
 $TITLE RAMP
 $ENTRY RAMP,5
"
"
 A $EQU 0
 B $EQU 1
 C $EQU 2
 K $EQU 3
 N $EQU 4
"
"
RAMP: INC N;FADD ZERO,ZERO
 SUB K,C;FADD
 MOV A,A;SETMA
 NOP
 MOV B,B;SETMA
 NOP
 DPX(0)<MD
 NOP
 DPY(0)<MD
"
"
LOOP: FMUL DPX(0),DPY(0)
 FMUL
 FMUL
 ADD K,C;SETMA;MI<FM
 DEC N;DPX(0)<FM
 BGT LOOP
 RETURN
 $END




 $TITLE GLITCH
 $ENTRY GLITCH,5
"
"  WRITTEN: R.W.HOBBS DECEMBER 1982
"
"  TO IDENTIFY AND SET TO ZERO UNWANTED DATA IN PRE-DEFINED WINDOWS
"
"  FORTRAN CODE:
" DO 100 I=1,NSAMP
" DO 200 J=1,NWINDOW
" IF(A(I).LE.0) GO TO 100
" IF(A(I).GT.B(2*J-1)) GO TO 100
" IF(A(I).LT.B(2*J)) GO TO 200
" A(I)=0.0
" 200 CONTINUE
" 100 CONTINUE
" RETURN
"
"
 A  $EQU 0  "BASE ADDRESS OF RAW DATA
 I $EQU 1  "ADDRESS INCREMENT
 B $EQU 2  "BASE ADDRESS OF WINDOWS
 COND $EQU 3  "NUMBER OF WINDOWS
 N $EQU 4  "NUMBER OF DATA POINTS
 TEMP $EQU 5
 BAD $EQU 17  "NUMBER OF ERRORS FOUND
"
"
GLITCH: INC N;FADD ZERO,ZERO
 SUB I,A;FADD
 LDDPA;DB=ZERO;DEC B
 LDSPI BAD;DB=ZERO
 MOVL COND,TEMP
"
LOAD: INC B;SETMA
 NOP
 DEC TEMP
```

```
 DPX(0)<MD;INCDPA;BGT LOAD
"
LOOP1: DEC N
 BEQ STOP;ADD I,A;SETMA
 LDDPA;DB=ZERO;MOV COND,TEMP
 INC TEMP
"
LOOP2: DEC TEMP;FSUB ZERO,MD
 BEQ LOOP1;FSUBR DPX(0),MD;INCDPA
 FSUB DPX(0),MD;INCDPA
 FADD;BFGE LOOP1
 BFGE LOOP1
 BFGE LOOP2
 MOV A,A;SETMA;MI<DB;DB=ZERO
 INC BAD;BR LOOP1
"
STOP: RETURN
 $END
```

THE TIME SEQUENCE ANALYSIS PROGRAM

1. TIME SEQUENCE ANALYSIS PROGRAM DESCRIPTION

2. DESCRIPTION OF PROGRAM OPTIONS

2.1 Time-Break and Source/Receiver Depth Corrections.

2.2 Trace Edit and Automatic Spike Detection Editing.

2.3 Mute Application.

2.4 Polarity Reversal.

2.5 Sea-Bed Consistant Statics on Unstacked Data.

2.6 Time Ramps.

2.7 Geometric Spreading Correction on Stacked Data.

2.8 AGC (Automatic Gain Control).

2.9 Band-Pass Filter.

2.10 Band-Reject Filter.

2.11 Wiener Spiking Filter.

2.12 Prediction Error Filter.

Notes on the Wiener and Prediction Error Filters.

2.12.1 How many filters can be applied

2.12.2 Choosing the autocorrelation function used for filter design.

2.12.3 Filter lengths

2.12.4 Prediction error filtering with prediction delays following the sea-bed.

2.13 Autocorrelation Function Output to Disc File.

2.14 Trace Normalization.

2.15 Reflection Strength Transform (Hilbert)

2.16 Normal Move-out Correction and Horizontal Stack

2.16.1 NMO and Stacking options

2.16.2 Velocity Profile Data

2.17 Three Trace Mix on Stacked Data

2.18 Velocity Filter on Unstacked Data

2.19 Dip Filter on Stacked Data

2.20 Time Varying Band-Pass Filter

2.21 Horizontal Sumnation of Stacked Data

2.22 Automatic CMP Consistant Statics

2.25 End of Processing and Output Trace To Tape

3. INTERACTIVE INPUT PROGRAM FOR PROCESSING PARAMETER FILE - PRCDAT.SAV

4. OPTIONAL DATA FILES DESCRIPTION

4.1 Stacking r.m.s. Velocity/Time Functions - DK6:VELRMS.DAT
Velocity Filter r.m.s. Velocity/Time Functions - DK6:VELFIL.DAT

4.2 Interactive Input Program for Velocity/Time Function Data Files - VELDAT.SAV

4.3 Sea-Floor Bathymetry Data File - DK6:DEPTH.DAT

5. THE PROGRAM PARAMETER FILE - DK6:SEIS.DAT

6. DURHAM SEG-Y REEL AND TRACE HEADER DESCRIPTION

1. TIME SEQUENCE ANALYSIS PROGRAM DESCRIPTION

The time series analysis program, often referred to simply as the "processing" program, applies selected signal processing operations to seismic traces input to the program. Time sequence analysis is a "tape-to-tape" operation in which each input trace in turn is read from one tape, is processed, and is output to a second tape. A schematic diagram of the program is shown in figs. I to III.

The processing program accepts input seismic data in Durham SEG-Y tape format only, see section 6 for details on the necessary reel and header block entries. An input common mid-point, CMP, gather may consist of any number of seismic data traces and auxillary data traces. Provided that the seismic data traces are sequential and form a contiguous block within the gather, also the total number of traces must remain constant for the whole profile processed. The user may define which seismic data traces to be included in the processing run, this permits the stripping off of a monitor record, the exclusion of auxillary traces or the windowing of a particlar offset range for a stacking operation. The input trace length may be upto 4096 samples and again must remain constant for the whole profile. The recording delay, the time between the shot instant and the start of the recorded data, must be constant within a CMP gather but may change along the profile. The output gather can follow the input delay function or may be forced to a given start time and trace length by padding with zero samples or truncating the input trace as necessary. The output trace length may be upto 4096 samples and are written to tape in Durham SEG-Y format.

The user determines which operations are performed on the input seismic traces by selecting entries from a "processing menu" and entering them in a program parameter file, DK6:SEIS.DAT. Table I lists the available processes. Section 2 presents detailed descriptions of each process; section 3 desribes the use of an interactive parameter input and checking routine PRCDAT, and section 5 lists how the parameters are stored for input to this program. The selected operations are performed in the order in which they are specified in the program parameter file. Up to 20 operations can be performed in one run of the program, most of the operations can be applied more than once in a run. Some of the options need extra data files, e.g. stacking velocity data, these are outlined in section 4, also some processes create temporary data files these are descibed where relevant.

Input data are assumed to have header blocks consistent with the Durham-SEG-Y tape format, i.e. headers as output by the Durham SORT program. The tape headers from the first input tape are stored in a disc file DK6:SEISHD.DAT and are updated when needed for output. Input trace headers are interrogated by the processing program and, depending on the operations selected, may be updated before output, see section 6 for more details. If the program is started part way through an input tape DK6:SEISHD.DAT will not be written so care must be taken to ensure the correct version is availiable.

The time series analysis program creates a simple log file, DK6:SEIS.LOG, which lists the updated tape header and any errors detected during processing. Also listed are the sequence number and the CMP gather number of the last trace processed before program termination. These numbers should be treated as a guide and be checked against the output tape, particularly when an error exit is made from the program when processing options include velocity filtering, dip filtering or three trace mixing.

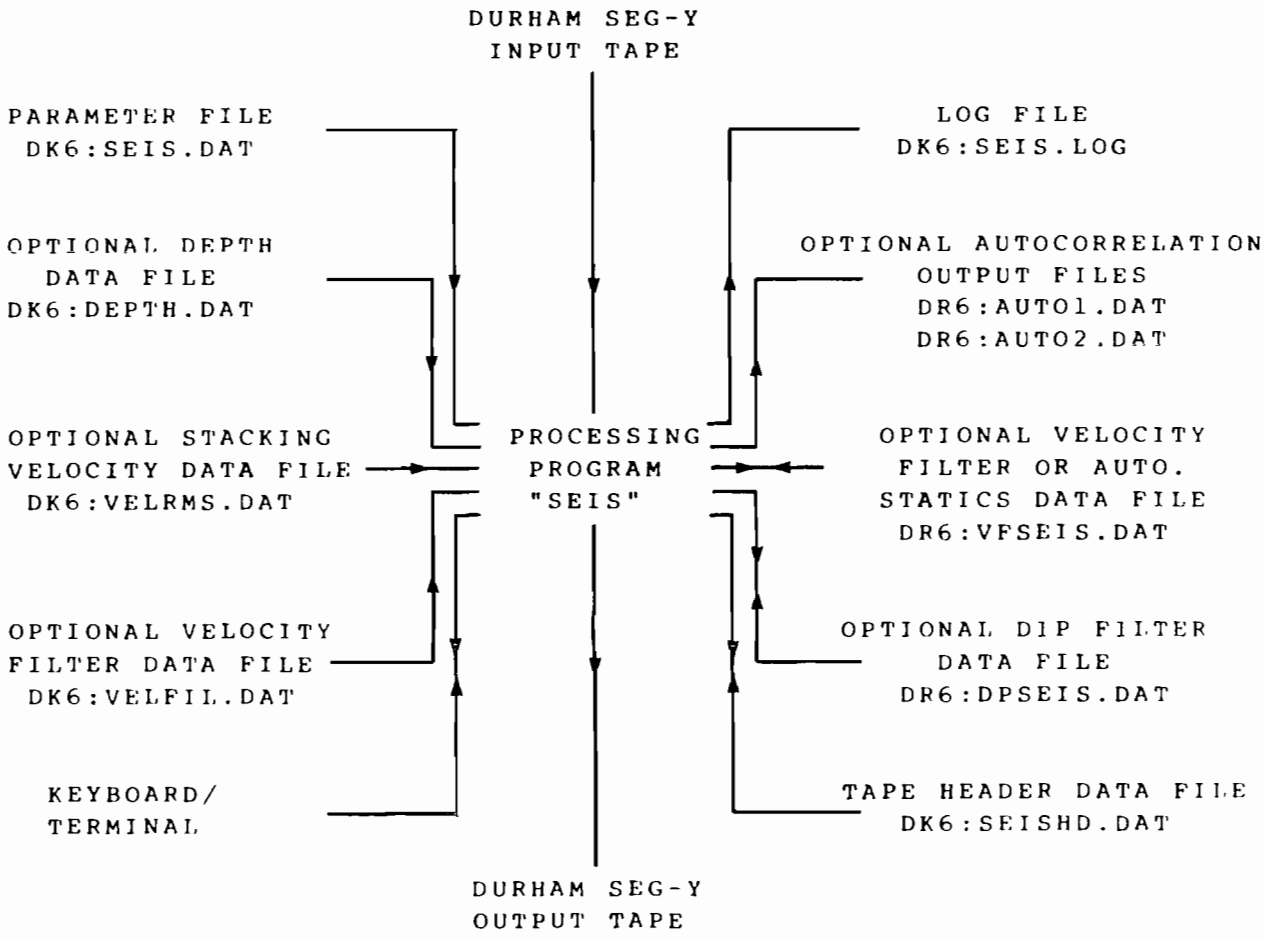


FIG I Input and Output Requirements

"NAME" Value	PROCESS
1	Time Break & Source/Receiver Depth Correction
2	Trace Edit & Automatic Spike Detection Edit
3	Mute Application (Start or End ofTraces)
4	Trace Polarity Reversal
5	Sea-Bed Consistant Statics on Unstacked Data
6	Time-Ramps (Application or Removal)
7	Geometric Spreading Correction (Application or Removal)
8	AGC Application
9	Band-Pass Filter
10	Band-Reject Filter
11	Weiner Spiking Filter
12	Prediction-Error Filter
13	Autocorrelation Function Output to Disc File
14	Trace Normalization
15	Reflection Stength Transform (Hilbert)
16	Normal Move-out Correction (Application or Removal) and/or Horizontal Stack
17	Three Trace Mix on Stacked Data
18	Velocity Filtering of Unstacked Data
19	Dip Filtering of Stacked Data
20	Time Varying Band-Pass Filter
21	Horizontal Sumnation of Stacked Data
22	Automatic CMP Consistant Statics
25	Exit and Output Trace to Tape

TABLE 1 THE PROCESSING MENU
Select operations in the order they are to be executed.
The Exit (25) must always be the last operation specified.

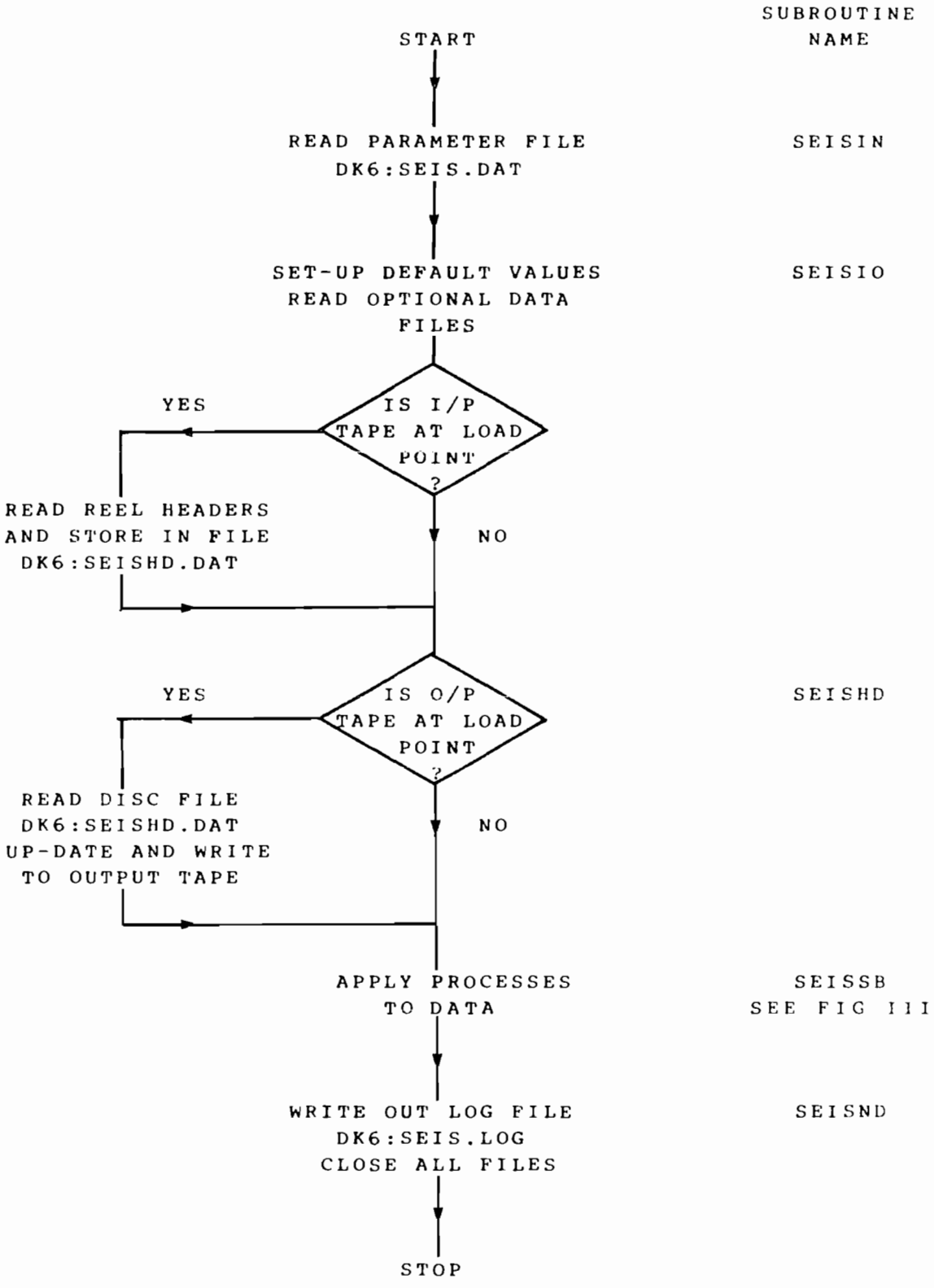


FIG II Flow Diagram for the Processing Program SEIS

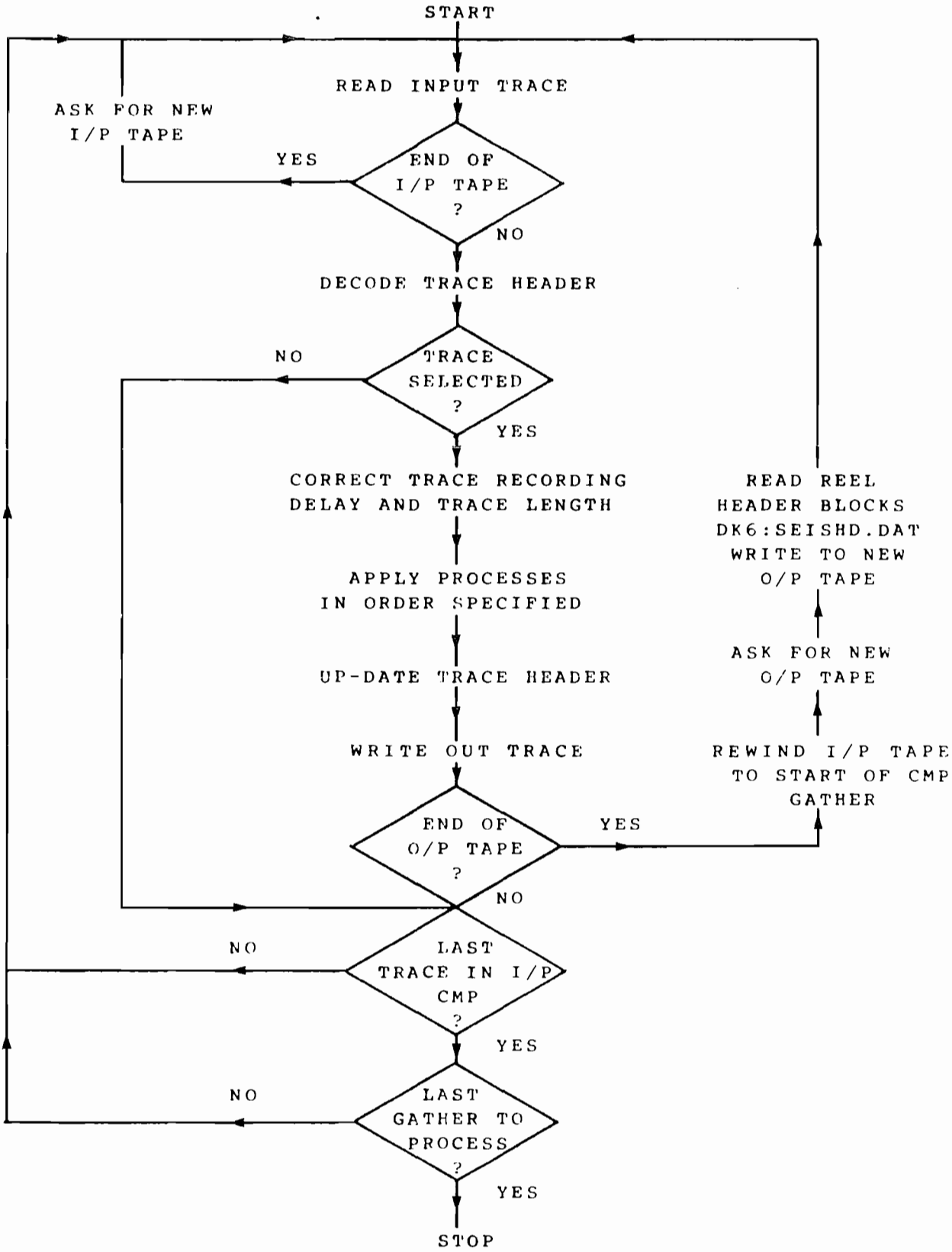


FIG III Data Processing in Subroutine SEISSB

2. PROCESSING OPTIONS

This chapter describes the function of each processing option and gives examples of its use.

Nomemclature

An input trace or an input CMP gather refers to data as stored on the input tape.

An output trace or an output CMP gather refers to data as stored on the output tape.

A processed trace or a processed CMP gather refers to data as stored in the program, after trace selection has chosen which input seismic traces are to be accepted for processing.

2.1 TIME-BREAK AND SOURCE RECEIVER DEPTH CORRECTIONS

MAY ONLY BE CALLED ONCE PER PROCESSING RUN

This facility moves the processed trace up or down to correct for the effect of the recording system "time-break" and for the fact that the source and receiver are submerged. The trace is truncated at one end and padded with zeros at the other so that the total length remains the same. With this correction applied, the trace appears as if the start of recording was exactly coincident with the shot - after allowing for recording delay - and as if the source and receiver were at the sea surface.

Be careful to distinguish the "time-break" from the recording delay. The "time-break" is a residual timing error inherent in most recording systems. Recording delay is the time from the shot instant to the start of the processed trace, e.g. in areas of deep water a recording delay of several seconds may be used.

Note that the "time-break" and the source/receiver depth corrections act in the opposite sense. A positive value of "time-break", recording starts before shot, pulls the trace up; positive values for source and receiver depth, submerged, pushes the trace down. The source depth and receiver depth function may be entered in the parameter file or may be taken from the individual trace headers, see section 6.

Although these corrections are often small they are desirable otherwise the velocities in the upper layers will be incorrect.

2.2 TRACE EDIT AND AUTOMATIC SPIKE DETECTION EDITING

MAY ONLY BE CALLED ONCE PER PROCESSING RUN

The trace editing facilities provide two methods of detecting bad traces and zeroing them:

(i) Particular traces which are known to need zeroing before processing begins can be specified explicitly. For each bad trace give the CMP gather number and the trace number within the processed gather.

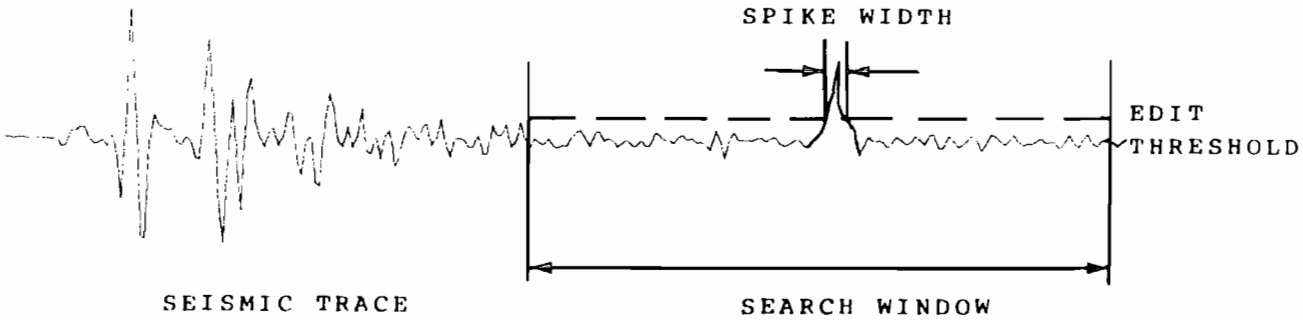
(ii) A section of the processed traces can be automatically examined by the program for the presence of noise spikes above a specified size, and if one is found, either the whole trace is zeroed or the offending sample may be set to zero.

If necessary, both these editing methods can be selected together in the same processing run.

When using the explicit trace editing facility, the program parameter file needs to be changed if the processing program is stopped and then re-started part-way through a line of data. Any entries for traces in the part of the line that has already been processed must be deleted.

Note that traces which are already dead on the input tape or not included in the processing run should not be entered for explicit editing. The processing program examines the header block of each trace before carrying out any other processing. "Dead" processed traces are skipped over (and if necessary a zero trace will be written to the output tape) without the editing instructions ever being reached. Including a "dead" input trace in the list of traces to be explicitly edited would result in subsequent entries in the list being read out of sequence.

The action of the automatic spike detection routine is illustrated below:



The program searches all processed traces across the defined window for any signal that exceeds the given spike amplitde threshold. If a spike width threshold is specified, i.e. greater than zero, and the spike has a duration longer than that given then the whole trace, not just the search window, is zeroed. If a spike width theshold is set to -1 then only the samples that exceed the amplitude theshold will be zeroed. Note that both positive and negative noise spikes are detected (although for clarity, only the positive case is shown in the above diagram).

Whichever editing method is used, if the whole trace is zeroed then the trace identification number is set to indicate a "dead" trace, see section 6. A message indicating which trace has been zeroed is written into the program log-file, DK6:SEIS.LOG.

2.3 MUTE APPLICATION

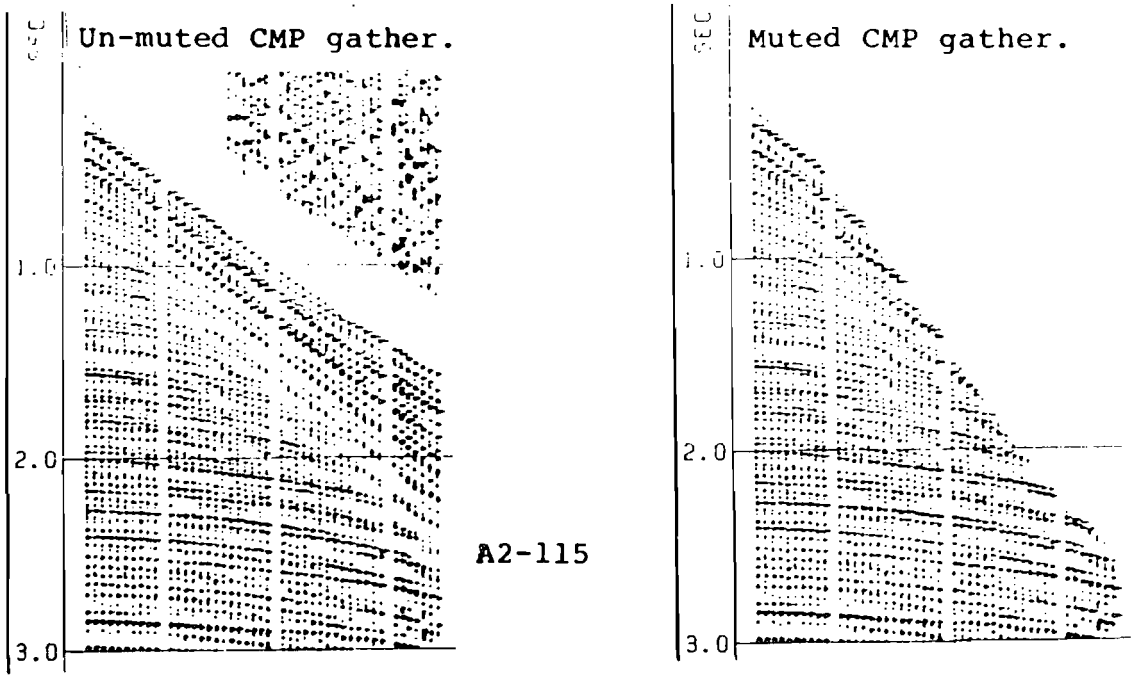
The mute function is used to remove coherent arrivals in the processed CMP gather that may interfere with the subsequent processing. For example; refracted arrivals at the start of the gather; wide angle reflections that will have greatly different phase charateristics from the near normal incidence arrivals and those arrivals that no longer conform to the hyperbolic approximation used when stacking the gather. The option may also be used as a post-stack process to remove noise from the sea-water column so that it will not appear when the trace has an AGC, see section 2.8, applied prior to display.

The muting facility enables a specified mute pattern to be applied to the traces of each processed CMP gather, the mute may be applied at the start or the end of the input traces. The edge of the mute has a cosine taper applied, the length of the which should be such that it is longer than the highest frequency contained in the data to avoid the introduction of suprious discontinuities in the data and hence aliasing problems in the frequency domain.

When applying a mute at the start of the traces, if the recording delay is longer than the absolute start time of the mute then the taper is applied to the start of the trace. If, when muting at the end of the trace, the absolute start time is greater than the total length of the trace, no mute is applied.

When a trace is muted, information about the applied mute is entered in the trace header block, only details of the muting at the start of the trace are recorded, since only this muting affects subsequent processing, see section 6. When stacking CMP gathers if a mute function has been applied the details must be present in the header for the correct normalisation to be applied, see section 2.16.1.

The function of the input parameter is illustrated in the following diagram, which shows the action of the mute on the start of a CMP gather:



2.4 POLARITY REVERSAL

MAY ONLY BE CALLED ONCE PER PROCESSING RUN

The polarity reversal facility enables particular traces in processed CMP gathers to be selected for polarity reversal.

If any traces within the CMP gathers have reversed polarity then subsequent stacking and velocity analysis will be incorrect.

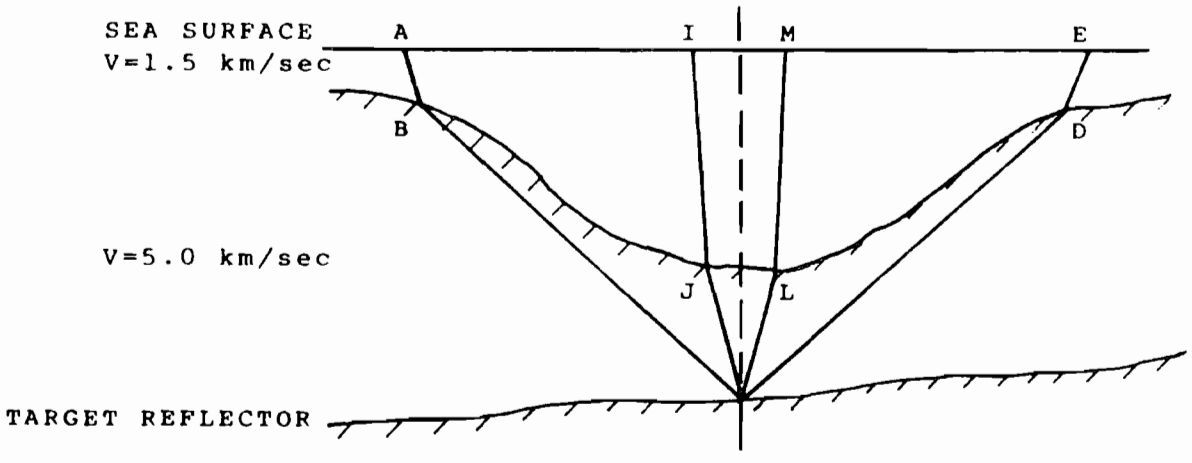
2.5 SEA-BED CONSISTANT STATICS

MAY ONLY BE APPLIED TO UNSTACKED DATA

The sea-bed statics routine corrects unstacked processed data for the effects of rapid variations in sea depth.

Changes in sea depth can cause significant effects when the sea-bed terrain varies over distances which are of the same order (or shorter than) the recording streamer length, particularly in shallow water. The following diagram illustrates this by showing ray paths to a reflector for the near and far offset traces of a CMP gather in a shallow trough. It is seen that the far offset trace has a much shorter path through the sea water (sections AB and DE) than the near offset trace (sections IJ and LM). If the rock head exposed on the sea-floor has a velocity of 5.0 km/sec this is much greater than the velocity of water at 1.5 km/sec. This results in a delay on the near offset trace similar to static corrections applied to seismic data aquired on land.

The sea-bed following routine uses the table of depth data values in DK6:DEPTH.DAT, see section 4.3, to calculate the sea depth under the shot, the receiver and the CMP points for each trace. The trace is then moved down or up as required to make it appear as if the depths at the source and receiver positions were equal to the depth at the CMP point.



2.6 TIME RAMPS

The application of a time ramp is one method of enhancing signal amplitude progressively down a trace to show up deep reflectors, also to help the seismic trace be closer to a stationary random process for deconvolution filtering. The time ramp facilities provide four types of time ramps which can be applied to and/or removed from each trace.

$$\exp^{(FACT*T)}$$

$$T*\exp^{(FACT*T)}$$

$$T^2*\exp^{(FACT*T)}$$

$$T^2$$

where: T is the time for the shot instant in seconds

$$N \text{ dB/SEC} = 20 \text{ LOG } (\exp^{FACT})$$

The ramp facilities can be called more than once in any one run of the processing program, allowing each trace to have a ramp applied, and after intermediate processing, removed.

2.7 GEOMETRIC SPREADING CORRECTION

MAY ONLY BE APPLIED TO STACKED DATA

The geometric spreading facility applies a TV^2 ramp to traces of stacked data only, to enhance signal amplitude increasingly down the trace. T is the absolute time (from shot) in seconds and V is the r.m.s. stacking velocity in km/sec. This gain profile is an alternative to the ramps described in the previous section.

As the program uses the velocity function placed in the trace header by the stack process, option 16, see section 6. Attempts to use the facility on unstacked input data cause the program to halt.

The geometric spreading facility can be called more than once per processing run, allowing each stacked trace to have a spreading correction applied, and after intermediate processing, removed.

2.8 AGC (Automatic Gain Control)

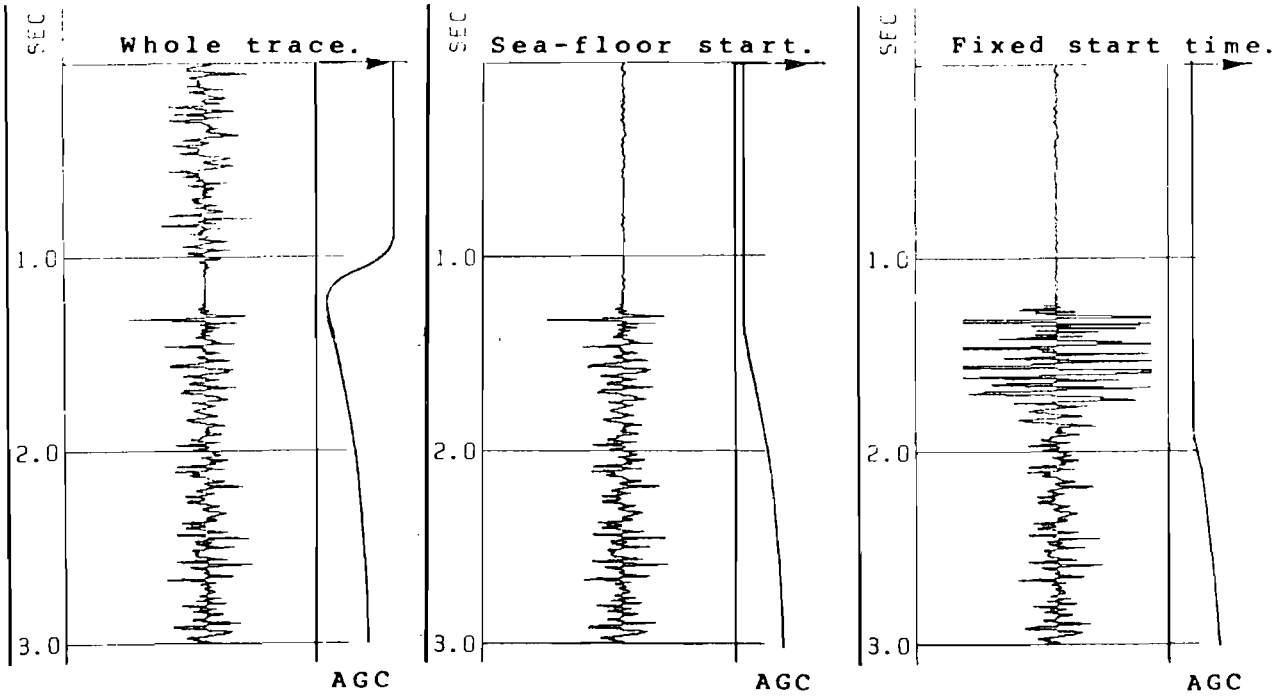
The AGC reduces the dynamic range of the signals along a trace in order to make the data more suitable for subsequent display plotting. Signal amplitudes are adjusted in such a way that the energy contained in a portion of the trace within the AGC averaging window is the same, irrespective of the position of the window along the trace.

Special considerations apply in the case of deep marine data. If the time to the sea-bed is longer than the AGC averaging window length, a conventional AGC will amplify the "sea noise" up to the same level as the real seismic signals further down the trace. To avoid this, the program can be instructed to apply the AGC only to that portion of the trace below the sea-bed, see section 4.3, when the averaging window is centered on the sea-bed. Another technique is to apply a mute to the sea-floor to remove noise in the water column, see section 2.3.

In cases where large amplitude signal, e.g. the sea-floor reflection, dominates the AGC function it is possible to fix a start time below the problem area. The gain calculated at the specified time is applied to the section of trace above this time.This does cause severe clipping of the large amplitude signal when plotting but avoids the "quiet" zone that occurs while the large amplitude arrival is in the AGC window.

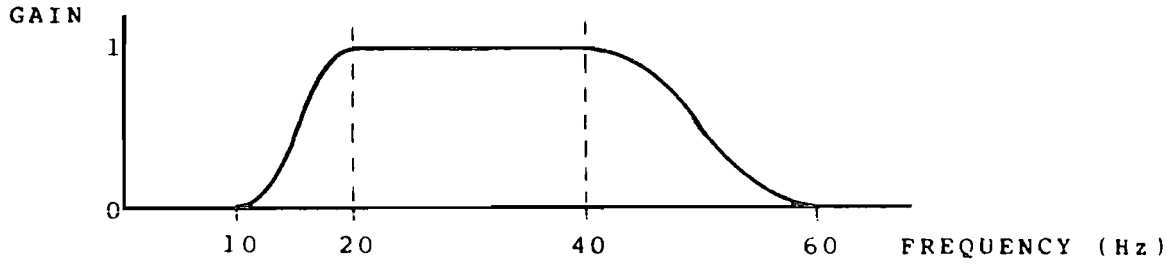
Note that once an AGC has been applied to traces, it is not possible to remove it with subsequent processing. Therefore this process is only used as a cosmetic feature prior to plotting.

The three examples show the effect of the AGC fuction on a whole trace; with sea-floor following; and with a fixed start time.



2.9 BAND-PASS FILTER

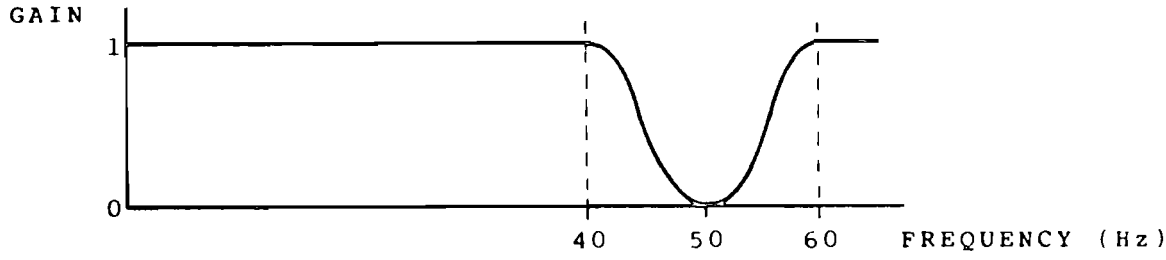
The band-pass filter removes frequency components outside a specified frequency window from all processed data traces. The boundaries of the frequency pass window are half-cosine tapered to avoid introducing "ringing" into the output data. The action of the band-pass filter is illustrated below. The filter is applied in the frequency domain. To prevent wrap round and edge effects the time series data has the first and last ten samples are tapered to zero before transformation.



2.10 BAND-REJECT FILTER

The band-reject filter removes frequency components of a specified frequency from all processed data traces. The rejection frequency is bounded by two half-cosine tapers to avoid introducing "ringing" into the data. The action of the band-reject filter is illustrated below. As in the band-pass filter described above the filter is applied in the frequency domain this necessitates the tapering to zero the first and last ten samples, this is done automatically by the program.

A typical application of the band-reject filter is to remove 50 Hz "mains hum" from data.



2.11 WEINER SPIKING FILTER

2.12 PREDICTION ERROR FILTER

NOTES ON THE WEINER & PREDICTION ERROR FILTERS

UP TO THREE DECONVOLUTION FILTERS MAY BE CALLED PER PROCESSING RUN

Although separate parameters are provided in the program data file for specifying a Weiner Filter and a Prediction Error Filter, the actual filtering process is executed by a section of program code that is common to both filters. (A Weiner Filter is the special case of a Prediction Error Filter with a prediction delay of one sampling period). It is appropriate, therefore, to describe the use of the two filters together.

2.12.1 How many filters can be applied

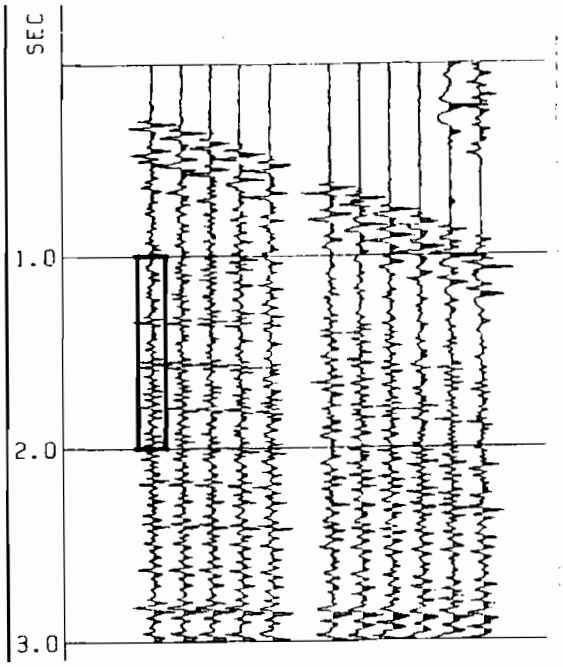
The processing program allows up to 3 filters to be applied in any one run of the processing program, and only one filter where the autocorrelation design flag is set to 0, see section 2.12.2. The first specified is the first applied, etc. For example, two Prediction Error Filters with different prediction delays can be applied in succession to the same data: one to remove a source bubble pulse and the other to remove a sea-bed multiple.

2.12.2 Choosing the Autocorrelation Function used for Filter Design

Designing a Weiner Filter or a Prediction Error Filter involves calculating the autocorrelation function using a window of the trace that is to be filtered. The section of trace which is to give rise to the autocorrelation function should be selected so that it spans several cycles of the periodic effect that the filter is to remove. The window is defined by three parameters, the design flag, described below, the absolute start time of the design window and window length. These determine which section of a trace, and which traces within a CMP gather, are used by the program to calculate the autocorrelation functions.

Autocorrelation Design Flag

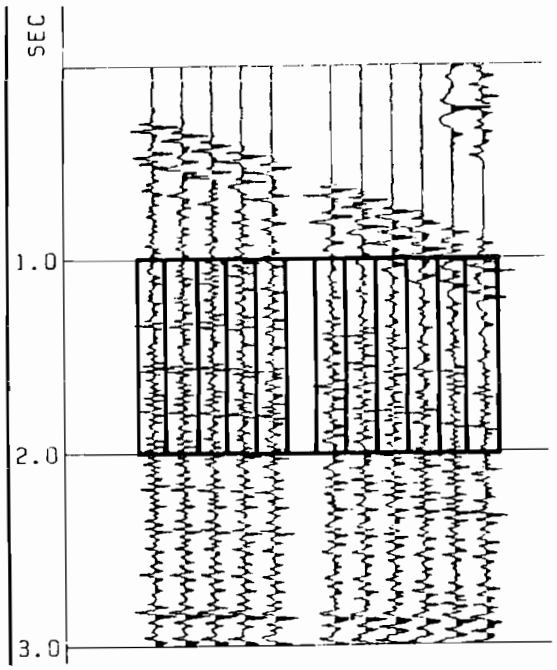
The value specified will depend on the type of data being filtered:



TIME SEQUENCE ANALYSIS PROGRAM

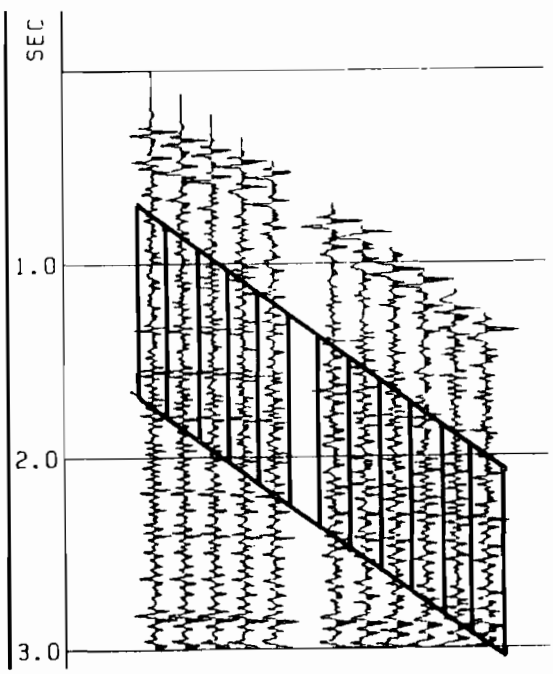
i) = 0

Is used for filtering unstacked CMP data where only the first non-zero seismic data trace nearest the ship is to be used for the design of the deconvolution operator for the rest of the gather. This may be required in cases where the signal to noise ratio on the far offset traces may be too poor for the design of an effective operator.



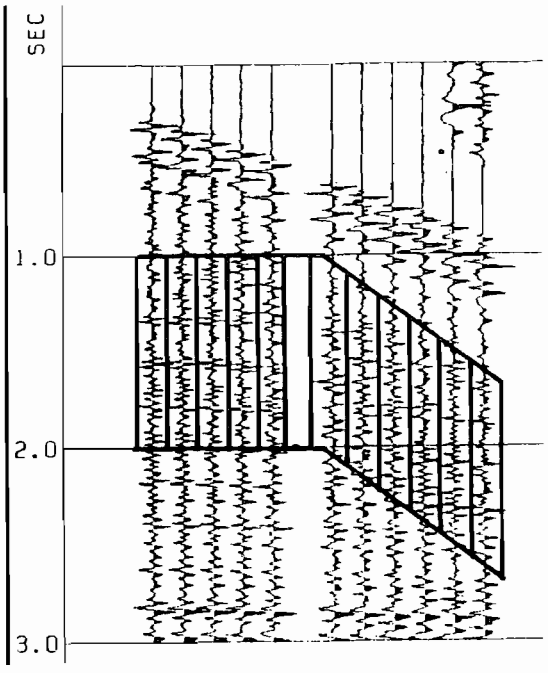
ii) = 1

Is used for normal deconvolution filtering work where a new operator is designed for each sesimic data trace. The start time and design window length for the autocorrelation function is the same for all traces.



iii) = 2

Is used for filtering unstacked CMP data to which a mute function has been previously applied. The design window length of the autocorrelation function is the same for all traces but the start time follows the mute pattern.



iv) = 3

Is used in special cases, such as the filtering of unstacked CMP data where the user needs to define all the autocorrelation design start windows to exclude refracted energy or wide-angle reflected energy which would distort the filter and which have not been previously muted. This option will only support upto 96 fold data.

2.12.3 Filter Lengths

For the Weiner Filter, the number of filter coefficients, n, is given by the filter length required multiplied by the sampling frequency. The number of filter coefficients calculated is one less than the filter length as the first coefficient, P_1, is always set to unity.

$$P_1 \; P_2 \; P_3 \; \ldots . P_n$$

For the Prediction Error Filter, the total number of coefficients is the sum of both the active filter length and the prediction delay, where n is the number of filter coefficients and m is the number of samples in the delay. The filter has the first coefficient set to unity followed by m-1 zeros, before calculated filter values.

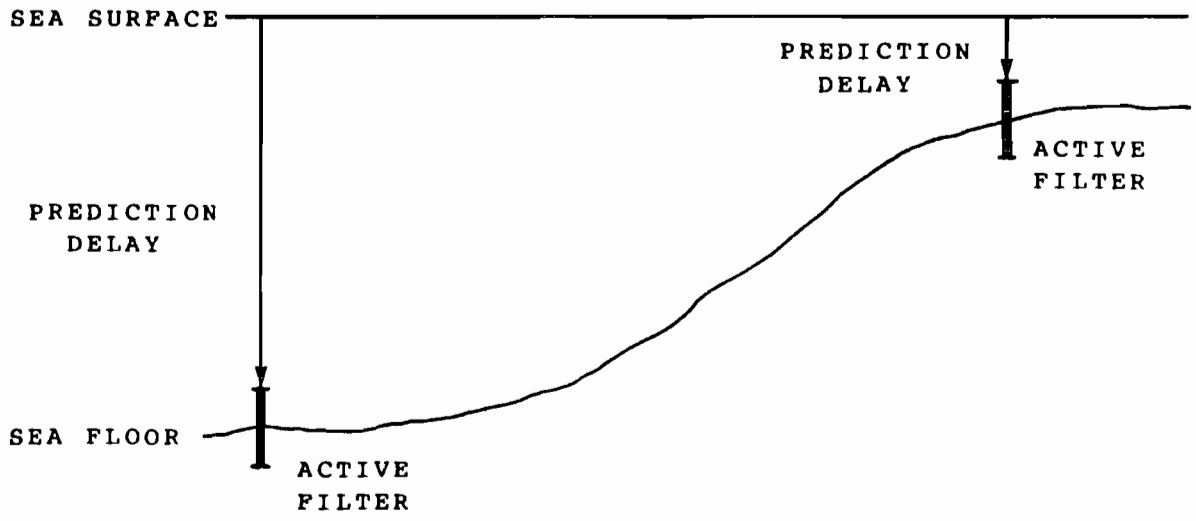
Prediction Delay	Prediction Filter
m-1 zeros	
1 0 0 0 0	$P_1 \; P_2 \; P_3 \; ..... \; P_n$

The Prediction Delay is chosen to be less than the period of the multiple to be suppressed.

2.12.4 Prediction Error Filtering with Prediction Delays following the Sea-Bed for Stacked Data

When using the Prediction Error Filter to suppress a multiple, the value of the prediction delay must be set to be less than the period of the multiple. However, in the case of a sea-bed multiple, the multiple period will change if the sea depth varies along the line. To overcome this problem, the processing program can be instructed to use the table of sea-depth values in DK6:DEPTH.DAT, or a value in the stacked trace header (see section 4.3), to calculate the sea depth at each trace position, and from this, evaluate an individual prediction delay for each trace.

The prediction lag value now represents the time in milliseconds before the sea-bed multiple that the prediction filter is to begin (see diagram below).



Prediction Error Filtering with Prediction Delay Calculated from Sea-Depth

2.13 AUTOCORRELATION FUNCTION OUTPUT TO DISC FILE

Examination of the autocorrelation function of a trace can often reveal periodic features which are not apparent from inspection of the trace itself. Hence, this is a technique used to determine the prediction delay needed to filter out multiples. Furthermore, comparison of the autocorrelation function of a trace calculated before filtering and calculated again after filtering, will show how successful the filter has been.

The processing program provides facilities to calculate autocorrelation functions and output them to a disc file for subsequent plotting. The position at which autocorrelation output is specified in the program data file determines which of the other processes selected are applied before the autocorrelation function is calculated, and which afterwards. The autocorrelation function output facility can be called up to 2 times in any one run of the processing program. For each trace processed, the first autocorrelation function is written into the disc file DR6:AUTO1.DAT and the second (if selected) is written into DR6:AUTO2.DAT. Each of these disc files will accommodate autocorrelation functions of up to 1000 traces.

The autocorrelation design options are similar to those described for the Wiener filter, see section 2.12.2, but the "0" option is not applicable and will give unpredictable results.

The program is arranged so that the length of an output autocorrelation function is always made equal to the length of the section of trace which gives rise to it. (This is different from the autocorrelation functions calculated and stored internally in the program during filter design). Thus, to look for hidden periodic features in a trace, specify an autocorrelation function constructed from the whole trace or a large part of it.

To plot the autocorrelation functions use the PLOT program. When specifying plotting gain, remember that the maximum value in the autocorrelation function has been normalised to unity and will occur at the zero lag element. The disc files also have a single header block attached which is copied from the first SEG-Y trace processed with the following changes; the number of samples is set equal to the window length multiplied by the sampling frequency; the trace indentification code is set to 9; and the number of traces per processed gather is also saved; see section 6.

2.14 TRACE NORMALIZATION

Normalization may be used instead of an AGC function, see section 2.8, to balance the trace energy or amplitude either within a CMP gather prior to stacking, see section 2.16, or as a cosmetic process on stacked data prior to plotting. Three options are available:

i)	-1	each input trace is normalized so that the largest amplitude is unity.
ii)	0	each trace is normalized to unit r.m.s. energy.
iii)	1	each trace is normalized by its mean energy.

2.15 REFLETION STRENGTH TRANSFORM (HILBERT)

The reflection strength transform produces an all positive low-passed filtered output where each sample amplitude represents the immediate signal strength at that point in the input data, the process is also called finding the envelope function. It is calculated by using the Hilbert Transform as shown below. This is a one-way transform that cannot be removed.

$$R_i = (a_i^2 + (H [a_i])^2)^{1/2}$$

for i = 1 to number of input samples.

If $a(t) \quad <-> \quad A(w) = R(w) + jX(w)$

then $H [a(t)] <-> H [A(w)] = X(w) - jR(w)$

2.16 NORMAL MOVE-OUT CORRECTION AND STACK

There are four options possible; to apply a normal move-out, NMO, correction to a proceesed gather and stack to form a single output trace for each CMP gather read; to take a processed CMP gather and apply an NMO correction without stacking; to remove an NMO correction from a processed CMP gather; lastly, to stack a previously NMO corrected CMP gather.

Option =	-2	remove NMO correction only
=	-1	apply NMO correction only
=	1	apply NMO correction and stack
=	2	stack only

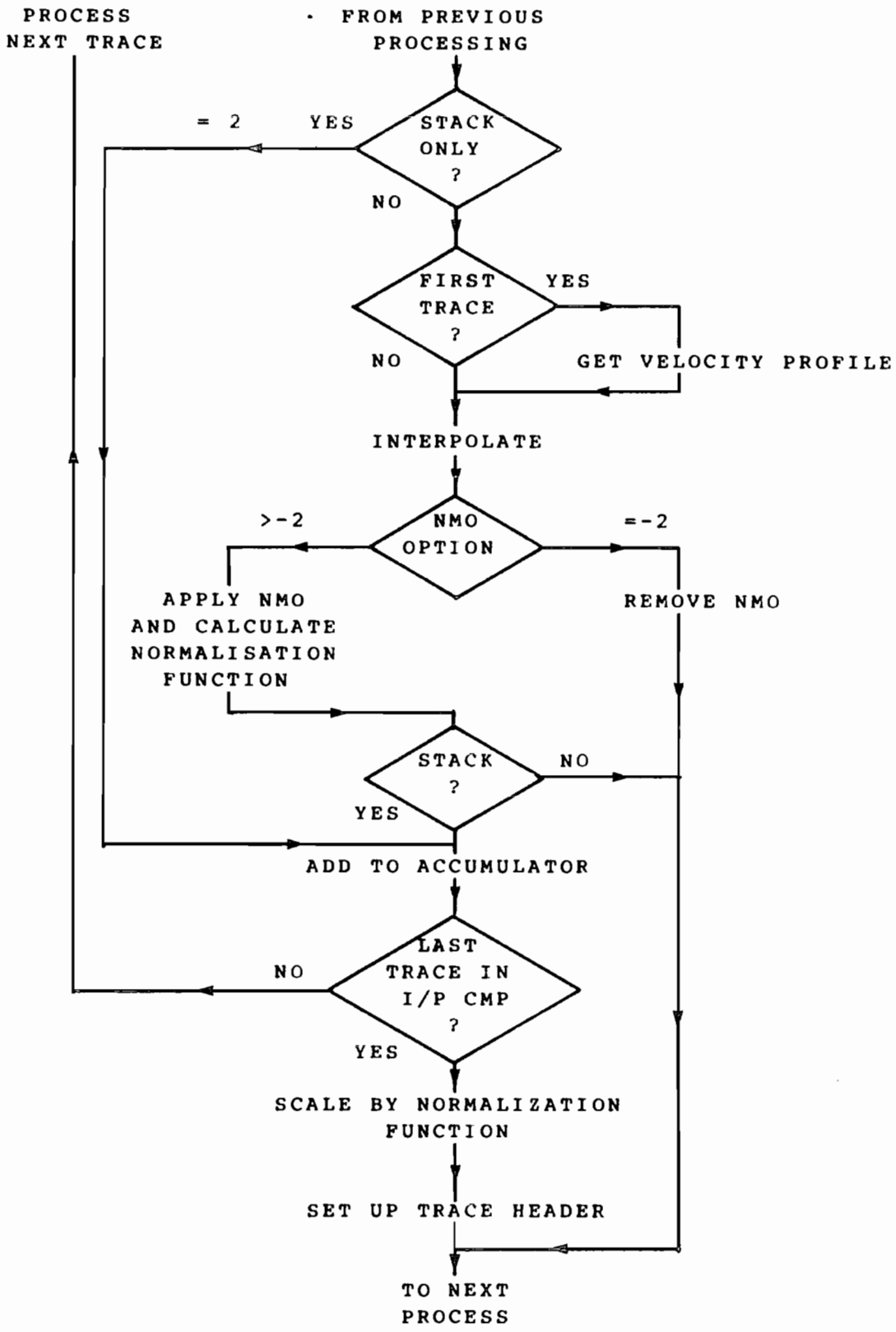
2.16.1 NMO AND STACKING OPTIONS

The full NMO and stack option is described as all the other options perform only part of this process.

The program requires that the traces in the input CMP gathers are presented with the first trace of the gather having least move-out.

The program takes the CMP number of the first trace within the gather and determines the appropriate velocity profile, see section 2.16.2. The r.m.s. velocity is assumed to be constant up to the first two-way travel time specified, then linearly interpolated between the other travel times entered in the table. This velocity function will then be used to calculate the NMO correction for each trace in that paticular CMP gather. The velocity profile is temporarily stored by the program, so that it can be written into the header of the stacked trace when it is eventually output to tape, see section 6.

A series of operations are performed on each trace of the processed CMP gather in turn. Firstly, sample interpolation prior to application of the NMO correction, this helps to reduce trace distortion during the NMO process. Next the routine calculates the intersection of the ideal move-out hyperbola with the input trace, then by linear interpolation of the input samples that fall either side the resultant moved-out sample is calculated. During the construction of the NMO corrected trace, the routine reduces the sampling frequency back to that of the original input trace. Finally, the corrected trace is added to a trace accumulator in which the stacked trace builds up as each of the component input traces is processed. This sequence of operations in illustrated below.



Flow Diagram for the NMO and Stack Process

Due to the limited array processor size high interpolation levels may not be possible with long traces. The processing parameters input program checks that the level of interpolation selected is compatible with the trace length and if not, automatically reduces it until it is. A message is displayed on the terminal to inform the user if this action is taken by the program, see section 3. Interpolation is carried out in the frequency domain and the frequency content of the interpolated trace remains the same as that of the original trace.

When all the traces of a CMP gather have been interpolated, NMO-corrected and added to the trace accumulator, the resultant stacked trace is scaled before being output to tape. To scale a stacked trace, the amplitude is divided by the fold of cover. Two different considerations which both affect the fold of over along a stacked trace (and hence also affect the trace scaling) are taken into account:

(i) The fold of cover is reduced if there were some blank traces in the CMP gather that has been stacked up. The program allows for this and so ensures that the amplitudes of traces in a complete stacked section are unaffected by short breaks in recording, or by any other effects that introduce small numbers of blank traces into unstacked data. The number of good (i.e. non-blank) input traces that have contributed to the stacked trace is stored in the trace header block, see section 6.

(ii) The fold of cover varies along a stacked trace if the traces of the input CMP gather from which the stacked trace originates were muted prior to stacking. The program takes account of this and so ensures that the scaling does not artificially reduce the amplitudes of signals near the top of a stacked trace. (This is important for shallow marine data where muting would otherwise lead to such a reduction in the amplitude of near-surface signals that the sea-bed would not be visible on the stacked trace).

If the option is -1, the normal move-out correction is applied, according to the velocity profile data (section 2.16.2), and the unstacked trace is passed on to the next processing option as defined in the parameter file.

If the option number is -2, then the move-out correction is removed from the input trace. This is done in a manner similar to that of normal move-out but the velocity funcion given, see section 2.16.2, has to be re-calculated as a velocity function at the required offset. This is done automatically by the reverse move-out routine.

For the case where CMP traces are moved-out using option -1, then after some intermediate processing the traces are stacked using option 2, the normalization coefficients used to compenste for pre-stack muting will be as for when the move-out was applied. It is therefore recommended that muting is applied prior to move-out correction rather than as an intermediate step.

There may only be one call to stack the data in any one run of the processing package, i.e. options 1 or 2. The options -1 or -2 options may be call more than once but only one velocity profile may be given, see section 2.16.2.

2.16.2 VELOCITY PROFILE DATA

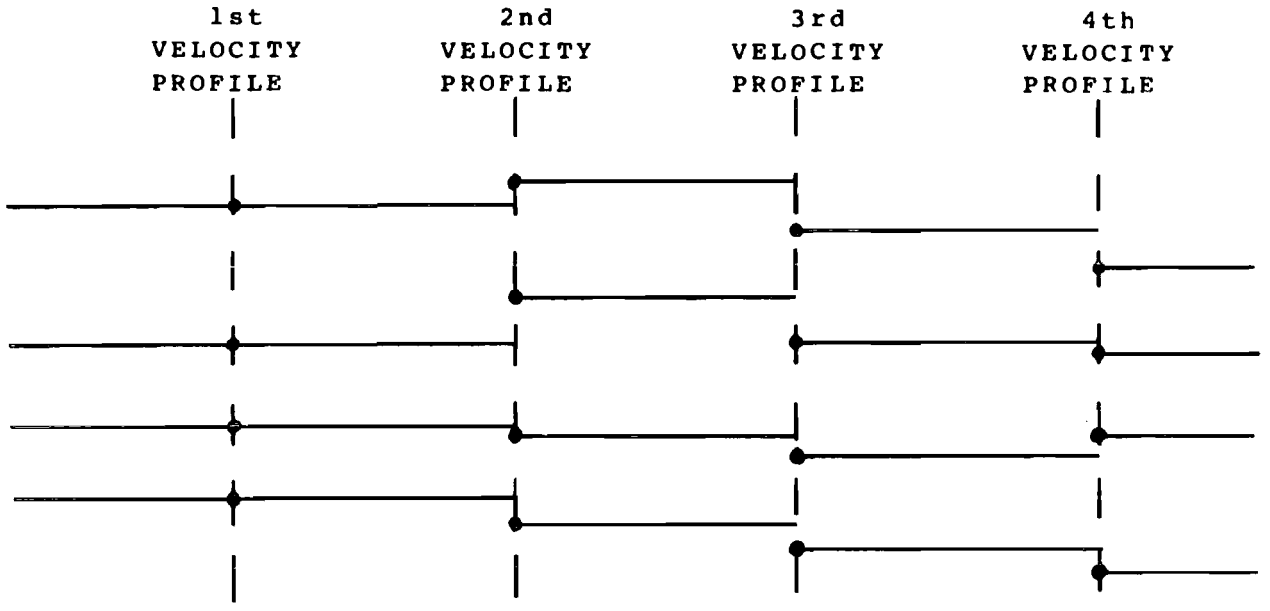
Successive velocity profiles along a line are specified in the stacking RMS velocity/time data file DK6:VELRMS.DAT by entering:

i) The CMP gather number of the velocity profile.
ii) The numbers of interfaces in the velocity profile.
iii) Pairs of two-way travel times and velocities; one pair for each interface.

See section 4.1 for further details of DK6:VELRMS.DAT data file entries, and the interactive input program, VELDAT.

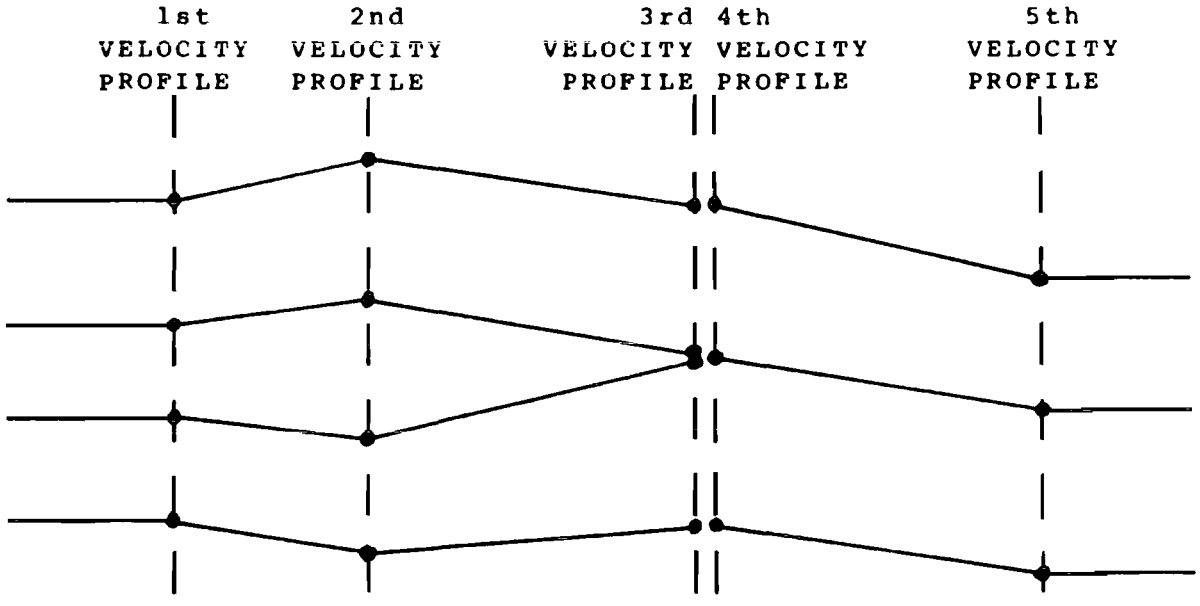
The routine must determine which of the velocity profiles specified in the velocity data file to use for each CMP gather stacked. The header block of the first trace in each input CMP gather is interrogated and the CMP number is used to determine the position of the gather along the line. The appropriate velocity profile can then be selected in one of two ways, depending whether lateral interpolation between velocity analyses is required or not.

If there is no lateral interpolation between velocity profiles along the line. All CMP gathers with CMP numbers between two successive velocity profiles are stacked using the velocity profile with the lower CMP number. At the start of the line, the first velocity profile in the program data file is applied to all preceding CMP gathers and at the end of the line, the last velocity profile is applied to all remaining gathers.



Otherwise the routine linearly interpolates between successive pairs of velocity profiles along the line and hence a new velocity profile is calculated for each CMP gather stacked. The beginning and end of a line, however, are handled in the same way as if no lateral interpolation is used, with the first velocity profile being applied to all preceding gathers and the last to all remaining gathers.

With lateral interpolated selected successive velocity profiles in the program data file must have the same number of interfaces, unless they are located at adjacent CMP points. Hence, to laterally interpolate across a "pinch out" requires the specification of two velocity profiles, one immediately before and one immediately after the "pinch out".



Program re-starts are handled completely automatically by the program. Unwanted velocity profiles are ignored. Thus, if the program is accidently stopped part-way through a line of data, it is not necessary to alter the DK6:VELRMS.DAT data file before re-starting the program.

The velocity profile used to stack each output trace is recorded in the output trace header block. See section 6 for details.

2.17 THREE TRACE MIX

MAY ONLY BE USED ON STACKED DATA

This is a crude dip filtering operation that may be used on stacked data to reduce section noise from non-spacially aliased dipping coherent arrivals, such as side swipe from off line diffractors. Each output trace represents the weighted sum of three adjacent input traces. The number of output traces equals the number of input traces.

$$a(t)_N = R(1)*a(t)_{N-1}+R(2)*a(t)_N+R(3)*a(t)_{N+1}$$

for N = 1 to number of processed CMP gathers

where R(i) is the weighting factor
a(t) is the stacked data trace

The data set to be processed is pre-padded and post-padded by a zero trace so that the total number input and output stacked traces are the same.

2.18 VELOCITY FILTERING

MAY ONLY BE USED ON UN-STACKED GATHERS

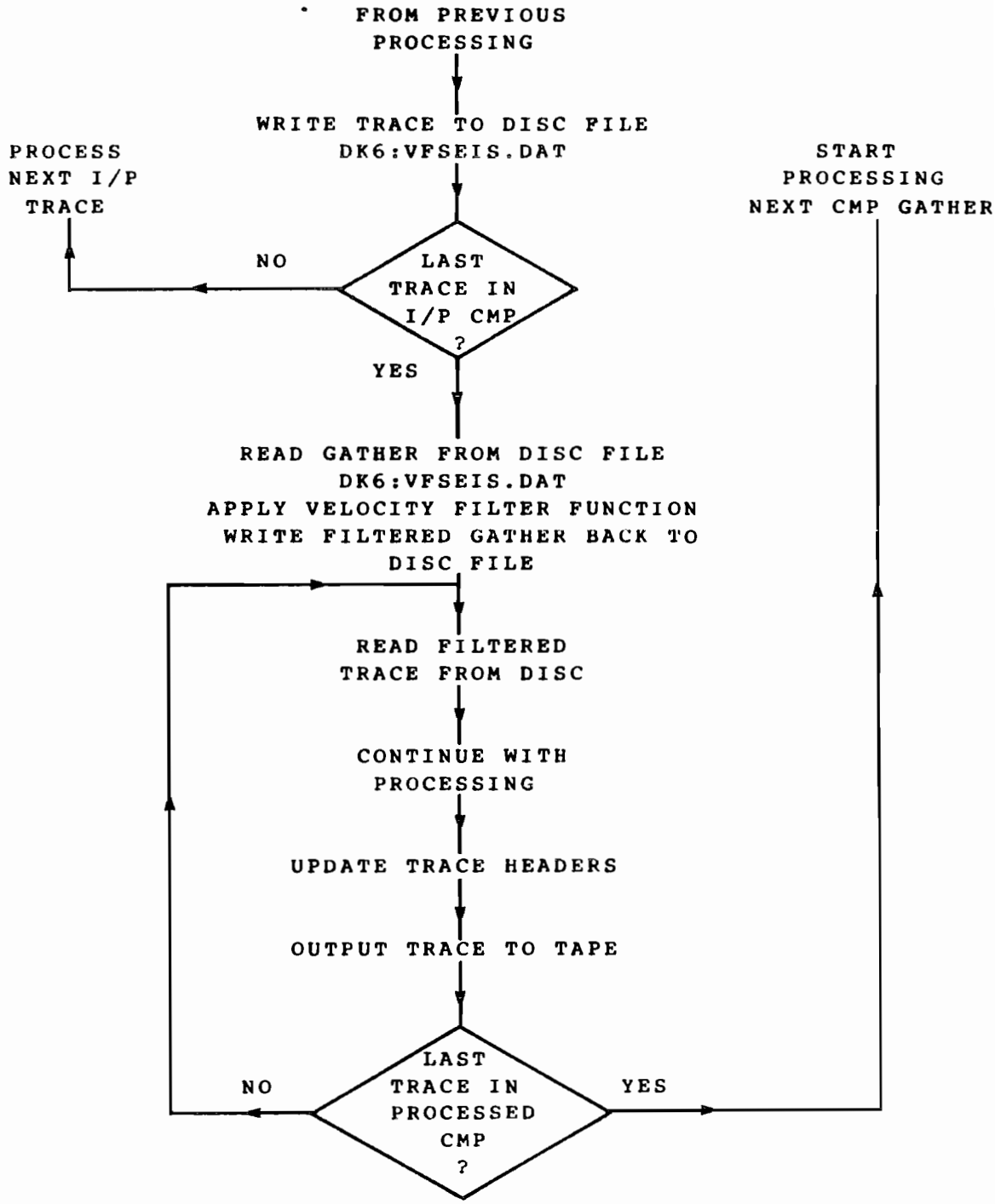
The velocity filter uses the technique (Ryu 1982) of applying a false NMO correction to the input gather then filtering using 2-D convolution with a simple operator, followed by the removal of the false NMO. This method is much more time efficient on small computers than transforming the whole gather into the frequency domain.

The size of the 2-D convolution operator determines the effectiveness of the filter, the value must be odd and less than 29, typically for twenty-four fold data a value of 11 is sufficient. The interactive input parameter program checks that the filter can be accomodated by the array processor, and automatically reduces the size if necessary. A warning message is given on the terminal, see section 3.

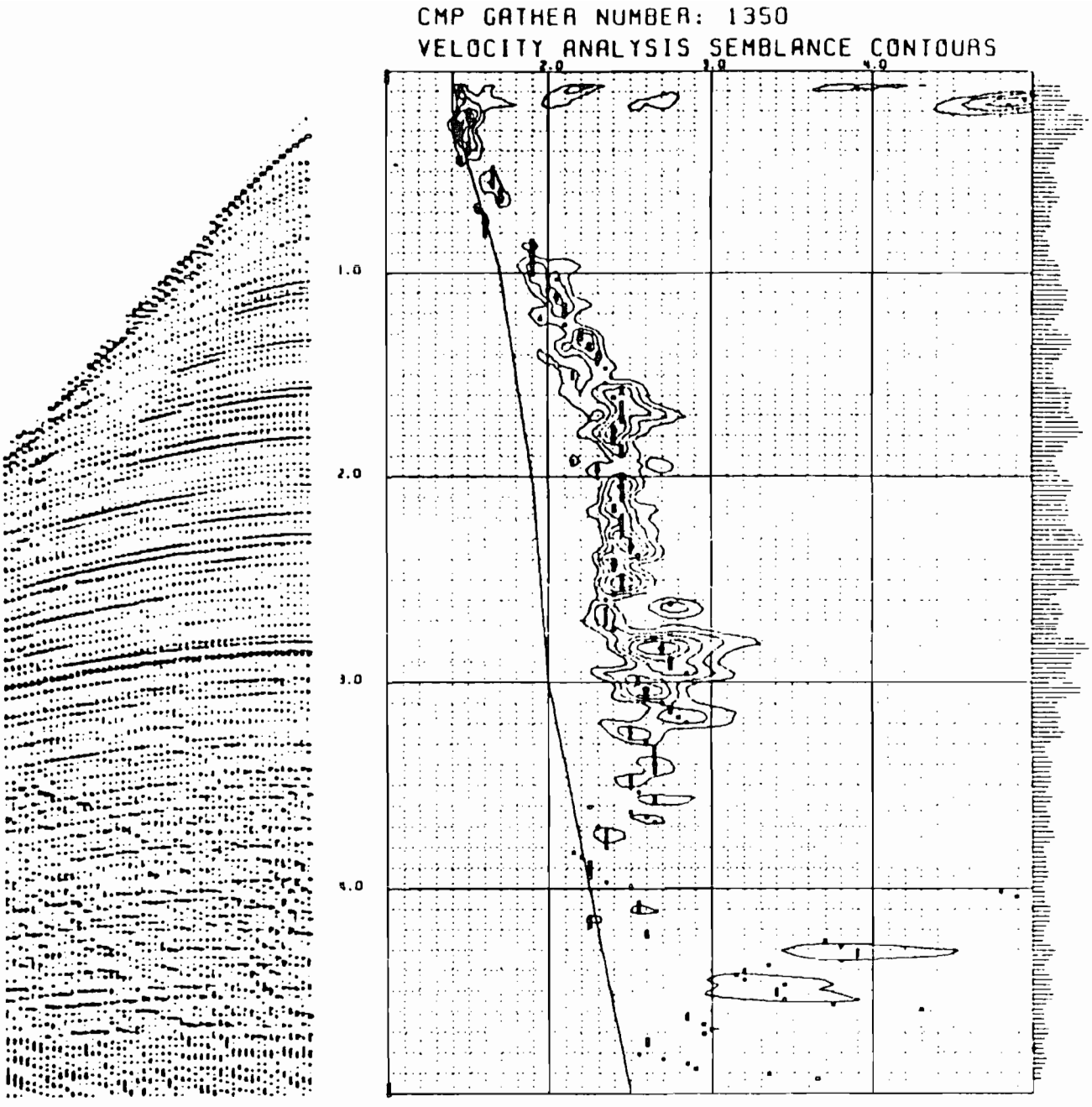
The velocity shift applied to the cut-off velocities allows the input velocity filter file to contain primary velocity picks. In this case a typical value of the shift should lie between 80% and 95%. If the file contains the required cut-off velocities then the shift value should be set to 100%. By setting a value of greater than 100% the filter reverses roll and removes coherent arrivals with an apparent stacking velocity higher than those specified in the velocity filter data file. The cut-off velocity information is contained in a data file DK6:VELFIL.DAT see section 4.1.

The filter generates its own internal calls to the normal move-out and stack option, section 2.16, to perform the application and removal of the false NMO correction. All required input traces are read, processed upto the velocity filter call and stored in a disc file, DR6:VFSEIS.DAT. The filter is then applied as a disc to disc operation with the filtered output traces occupying the same locations as the input traces. Subsequent processing requests then read the seismic data from this disk file. The file is written in a pseudo-SEG-Y format that is incompatible with disc files written for velocity analysis and plotting routines.

Care should be exercised when choosing the cut-off velocity function as it is possible to filter out primary data along with the rejected noise. It has been found that the best cut- off velocity function is one that lies equi-distant between the best stacking velocity function and the low velocity multiples, as shown below.



Flow Diagram for the Velocity Filtering Process



Semblance Plot showing Optimum CutOff Velocity Fuction

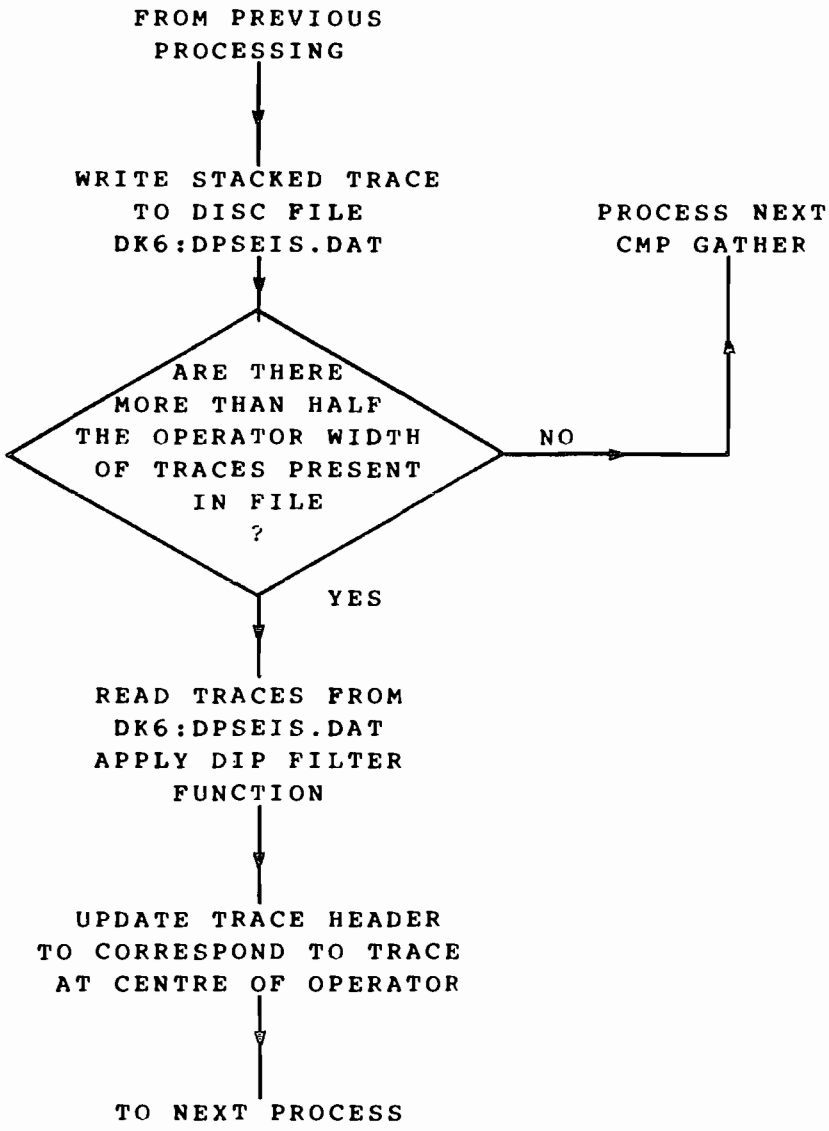
2.19 DIP FILTER

MAY ONLY BE USED ON STACKED DATA

A simple fan filter is designed in the frequency domain according to the given apparent velocity parameter. It is transformed into the x,t domain for application, as this is more time efficient for small computers. The stacked seismic traces that are to be used by the filter for a given output trace are stored in a pseudo-SEG-Y disc file DR6:DPSEIS.DAT. This file is in an internal format for use by the Time Sequence Analysis package only and may not be used as input for any other program.

The choice of the minimum apparent velocity to be passed can be calculated from the steepest slope arrival that is to be retained in the final section. The filter will totally reject all coherent arrivals with more that twice the choosen slope, arrivals that lie between these two limits will be suppressed according to a cosine taper.

This is a more versatile filter than the Three Trace Mix, section 2.17, but takes longer to execute. The stacked data is automatically pre padded and post padded with zero traces so that the total number of traces remains the same.



Flow Diagram for the Dip Filtering Process

2.20 TIME VARYING BAND-PASS FILTER

The routine uses a Butterworth band-pass filter characteristic, where the lower and upper frequencies describe the -6 dB points of the response curve, i.e. the pass-band. The filter is applied using infinite impulse response recursion techniques (Nikolic 1975).

For times before the first specified and after the last, the frequency response remains constant at the given parameters. For time between the two limits both the upper and lower cut-off frequencies change linearly.

For profiles along which the bathymetry varies greatly, it is better to fix the band-pass characteristics relative to the sea-floor. In this case the time represents the delay after the sea-floor for the specified band-pass characteristic, see section 4.3.

2.21 HORIZONTAL SUMNATION OF STACKED DATA

MAY ONLY BE USED ON STACKED DATA

An alternative to dip filtering, see section 2.19, or trace mixing, see section 2.17, is to horizontally sum several ajacent stacked CMP gathers together to form a single output trace. This effectively increases the fold of cover at each output stacked CMP gather by the number of input traces summed but increases the CMP spacing by an equal ammount. This process is mainly used to contract large data sets where a split sort has been performed, i.e. the receiver spacing is less than the twice the shot spacing, back to the original shot spacing.

2.22 AUTOMATIC CMP CONSISTANT STATICS

MAY ONLY BE USED ON UNSTACKED DATA

Due to unknown errors within each CMP gather, such as the receiver array not being at a constant depth, see section 2.1, or slightly incorrect gross static corrections, see section 2.5. The trajectory of a given reflector in the CMP gather may not be hyperbolic, assuming that the offsets and velocities involved allow the first order approximation to be made. This option enables small corrections to be applied within the gather with respect to a pilot trace. The process is more fully desribed in an M.Sc. dissertation by Loveridge (Durham 1981).

Prior to calling this routine it is necessary to apply the correct normal move-out correction, see section 2.16, without stacking the gather. The data is then stored in a disc file, DR6:VFSEIS.DAT the same file is used for the velocity filter routine, see section 2.18, but the two processes do not interfere

even if called in the same processing run. The pilot trace may be formed from a single trace within the proceesed gather or a brute stacked group of traces, each processed trace is then cross correlated with the pilot trace over a specified window. The resulting cross correlation function is scanned for its maximum value and its' offset from the zero lag coefficient is calculated, this number represents, in samples, the shift necessary to optimally line up the prominent reflectors in the specified window. Further, the amount of shift applied may be limited to avoid leg skipping, i.e. shifting the processed trace by more than half a wave-length. The whole process is repeated twice with a new pilot trace calculated from first attempt as this was found to give a more stable correction. The normal move-out correction may then be removed using option 16 prior to further processing.

It is advised that care should be taken if a mute function is applied before the auto-statics that the cross correlation window does not extend into the moved-out muted region otherwise unpredictable results may occur.

2.25 EXIT AND OUTPUT TO TAPE

ALWAYS CALLED AS THE FINAL PROCESSING OPTION

This indicates to the program that all processing has finished on the given trace and to output the seismic data plus the updated SEG-Y trace header to the specified output tape.

3. INTERACTIVE INPUT PROGRAM FOR THE PROCESSING PARAMETER FILE DK6:SEIS.DAT

The interactive input program performs two functions; to provide a user friendly method of entering processing parameters; and to act as a parameter checking routine on a parameter file created or modified by the computer EDIT program.

It is essential that all parameter files pass through this program as it contains all the parameter checking and default set-up routines. These are not repeated in the main processing package.

To start the program type:

```
RUN DK1:PRCDAT
```

The program will respond with a title and a question, for interactive mode type a carriage return.

Respond to the questions asked following the format indicated; integer, no decimal point allowed; real, decimal points will be accepted; (Y/N) or similar, expect one of the letter options to be returned. A "?" prompt indicates how many entries are expected on that line, if more than one, seperate entries by a comma. In the case where lists of numbers are to be given a prompt '12 per line' is given, enter numbers in lines of 12 each seperated by a comma, finish each line or part line with a carriage return.

The first series of questions are to define the input data tapes and the total number of files to process in this run. This number may be much larger than the number of files in the profile as the program may be stopped at a read or write tape change.

The second part sets up which part of the input gather to include in the processing run in both trace sequence number and time window.

The read and write tape drive numbers instruct the program where to input and output the data respectively. The reel number is a parameter entered into the ASCII reel header block a gives the sequence number of the output reel within the whole profile processed, this number is incremented automatically by the program at each write tape change. The trace sequence number gives the absolute position of each trace within the output seismic data set, normally starting at 1 at the start of the profile and incrementing by one for each trace output. If the program is stopped part way through a profile, when re-starting this number should be changed to represent the next trace sequence number to be written.

A processing menu is now given from which the required processing option may be made. It is important to enter the options in the order that they are to be applied onto the input gather. There is no facility for the interactive editing of parameter files that need changing or have mistakes in them. In this case either restart the program or use the EDIT program. After each option is selected the program will prompt for any necessary information for the given process. The user is then returned back to the processing menu for the next option. The responses to each parameter entered are checked, default values are inserted as neccessary and warning messages are given if any value is modified by the program to comply with the space limit imposed by the system.

To finish the processing sequence enter option 25, End of Processing and Output Trace to Tape. The program asks if the processing options choosen are correct, if not, no parameter file will be written and the program terminates. Text lines may now be given, these will appear in the ASCII reel header and the processing log file DK6:SEIS.LOG, along with the current date. These are for information only and take no part in the processing of the data.

For the checking of a parameter file previously created by the EDIT program, or one that has been modified. Instead of entering a carriage return as the response to the first question, give the full file name of the input parameter file, e.g.

..... ? DK6:LINE6.DAT

The program will read the file, and as above the responses to each parameter entered are checked, default values are inserted as neccessary. The program lists all the options called and warning messages are given if any value is modified by the program to comply with the space limit imposed by the system. The values of the start reel number and trace sequence number are updated with new values entered from the terminal.

The program keeps a list of any auxillary data files needed for the given processing options. These are listed at the end as a reminder to ensure that these files are available, see section 4.

The processing parameter file is always written DK6:SEIS.DAT as this allows the direct tranfer of control from the input program to the processing program. If the processing program is not to be run immediately, then the parameter file is left on DK6: and the user is returned to the operating system. The file may be renamed as a precution to overwriting but prior to the execution of the processing package must be copied back into DK6:SEIS.DAT, this may be done by running PRCDAT.

4 OPTIONAL DATA FILES

4.1 VELOCITY/TIME PROFILES

If the processing option 16, NMO and stack, is requested the program will expect to find a file DK6:VELRMS.DAT. This will contain the r.m.s. stacking velocities and two-way travel time of the primary reflectors taken from the velocity analysis semblance plots.

When velocity filtering, option 18, the cut-off velocity function must be present in a extra data file DK6:VELFIL.DAT. If this file contains primary velocity picks from the velocity analysis semblance plots then the velocity shift value, see section 18, should be in the range 80% to 95%. If the file contains a cut-off velocity function that lies between the primary events and the multiple events then use a velocity shift of 100%.

Both these files have the following structure:

```
VELOCITY ANALYSIS 1
VELOCITY ANALYSIS 2
    .       .
    .       .
```

The velocity analyses are entered in CMP order and have the following form:

```
CMP NUMBER,  NUMBER OF LAYERS
TIME(1),  VELOCITY(1)
TIME(2),  VELOCITY(2)
  .           .
  .           .
```

CMP NUMBER (I6) Common mid-point number where given velocity function is to apply.

NUMBER OF LAYERS (I6) The number of velocity/time picks in the given analysis. Upto 10 layers may be entered.

TIME (1-N) (F10.0) The two-way travel time in milliseconds to a required pick.

VELOCITY (1-N) (F10.0) The r.m.s. velocity in kilometres/second at the given time.

4.2 INTERACTIVE INPUT PROGRAM FOR VELOCITY DATA FILES - VELDAT

This program allows full interactive editting of a r.m.s. velocity data file. The use of which is highly recommended to ensure the correct entery of the velocity/time information.

To start program type:

```
RUN DK1:VELDAT
```

The program will ask if an existing data file is to be re-editted, if so, having entered the full name, the file is read, then renamed to *:*.OLD . The output will be written to the the named file. If starting a new file only the output file is created to which all the entered data is written.

The editting options:

INSERT A CMP	Allows the input of a new velocity function at the CMP given. If a velocity function already exists for the specified CMP the program will go into CHANGE mode, see below.
DELETE A CMP	This deletes the whole CMP velocity funtion from the data file.
CHANGE A CMP	To change any given r.m.s. velocity/time pair from within a given CMP velocity function. There are four actions possible: i) Insert an extra layer. ii) Delete a layer. iii) Change a layer. iv) Accept CMP, return to basic menu.
EXAMINE A CMP	Lists the given CMP on the terminal, giving both r.m.s. velocities and associated calculated interval velocities.
PRINT	This outputs the full file to the printer with headings, times, r.m.s. velocities and calculated interval velocities.
VERIFY LAYER NUMBER	For the case where lateral interpolation is required by the processing program, see section 2.16.2. This option checks that any layer number changes occur at adjacent CMP's.
LIST CMP NUMBERS	Lists on the terminal all the current CMP numbers where velocity functions have been entered in the file.
EXIT	Terminates the program and writes out the file to the name specified.

The program automatically orders the velocity analyses present at any one time. It checks for non-real interval velocities, i.e. the Dix formula gives a negative number for the square of the interval velocity. In this case an error message is sent to the terminal.

When entering time/velocity information it may be specified in either of two ways. Firstly as a two-way travel time in milliseconds and r.m.s. velocity in km/sec seperated by a single comma, e.g. 2000,2.5 ; or, as two-way travel time and interval velocity in km/sec seperated by two commas, e.g. 200,,4.0 . The program requires that the velocity/time pairs be entered in time increasing order. Failure to do this will cause the program to abort the current function and return to the basic menu.

In the case of stopping the processing of a line of data part way through and subsequently resuming from the halt point, it is not necessary to edit out the velocity analyses already used.

4.3 DEPTH DATA FILE DK6:DEPTH.DAT

Echo sounder depth measurements are often made at regular distances along a line. Usually, however, the echo soundings are much less frequent than the seismic shot interval and it is therefore necessary to interpolate the echo sounder data to determine the sea depth at particular shot points.

The depth data file is simply a list of echo sounder depth measurements preceeded by time and ship/streamer geometry information that enables the program to determine the correspondence between the positions of the echo soundings and the positions of the seismic source, the receivers and the CMP points.

The depth data file must be present if when pre-stack processing the sea floor consistant statics, option 5, is requested.

When post stack processing the sea-water depth may still be taken from the DK6:DEPTH.DAT file or if this is not present the program will take the first two-way travel time from the time/stacking velocity table stored in the stacked trace header, see section 6. The following post-stack options use sea-floor depth information:

i) Mute to sea-floor
ii) AGC starting at sea-floor
iii) Prediction Error Filter with prediction delay following sea-bed.
iv) Time Varying Band-pass Filter with the changing band-pass characteristics fixed relative to the sea-floor.

FORMAT OF DEPTH DATA FILE - DK6:DEPTH.DAT

```
IYEAR, IDAY, IHOUR, IMIN, ISEC
ISSINT, IXES, IXSR, IXSS, IXRR
ISSHOT
IDPINT
IDPTOT
IDEPTH (1)
IDEPTH (2)
     .
     .
IDEPTH (IDPTOT)
```

Where:

IYEAR	(I6)	Year - e.g. 1983	
IDAY	(I6)	Julian day	Time of first depth
IHOUR	(I6)	Hour	measurment in file.
IMIN	(I6)	Minute	
ISEC	(I6)	Second	

ISSINT (I6) Shot interval in SECONDS (not milliseconds)

IXES (I6) Distance between echo sounder and source, in metres.

IXSR (I6) Source - Receiver offset, in metres.

IXSS (I6) Source spacing - distance between the positions of successive shots, in metres.

IXRR (I6) Receiver spacing, in metres

ISSHOT (I6) Number of the most recent shot fired at the time when the first depth value in the file was measured.

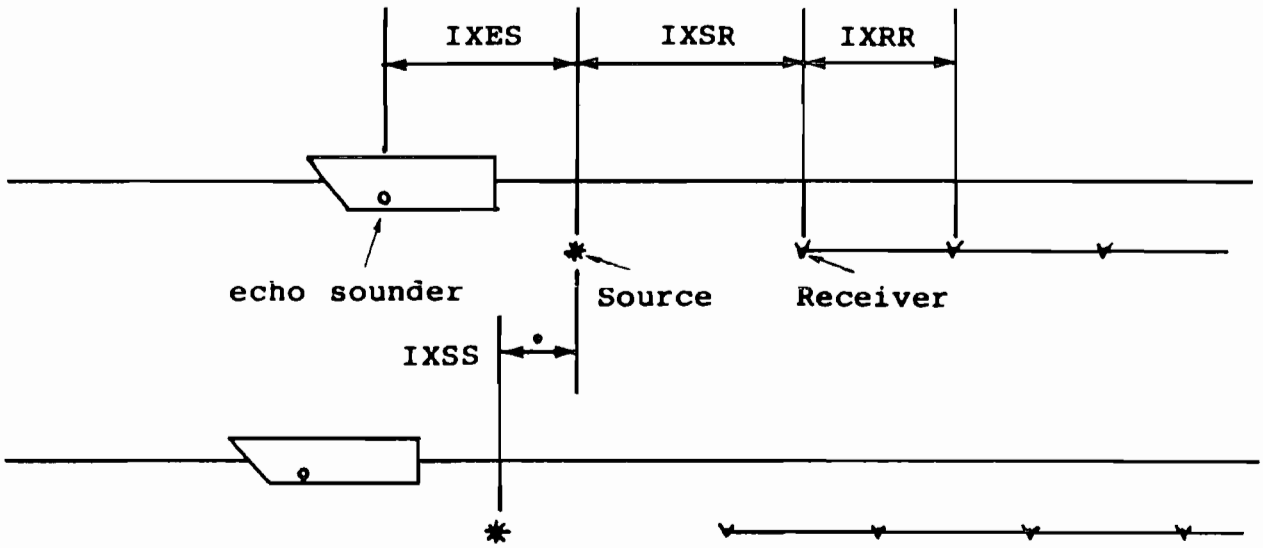
IDPINT (I6) Time interval in seconds (not milliseconds) between successive echo-sounding depth measurements.

IDPTOT (I6) Total number of depth measurements in file.

IDEPTH (1-IDPTOT) (I6)
Depth values in metres - 1 value per line. These are ONE-WAY depths and, strictly, should be corrected to sea-surface to take account of the fact that the echo-sounder is at a finite depth in the water. (However, the time-break correction can be employed if only uncorrected depth values are available).

In the case of halting the processing of a line of data part way through and subsequently resuming from the halt point, it is not necessary to edit out the depth values already used.

In the case where depth data values are available only for part of a line, the first depth measurement is applied to all preceeding seismic traces, and the last depth measurement is applied to all subsequent seismic traces.



The position of ship and streamer at two successive depth measurement positions, show meaning of the geometry parameters. An echo sounding depth measurement is made once every IDPINT seconds. The seismic source if fired once every ISSINT seconds (the shot interval).

5. THE PROGRAM PARAMETER FILE - DK6:SEIS.DAT

```
NFILES, NCHAN, NSAMP, FSAMP
IFIRST, ILAST, ISTART, ILEN
TPDRR, TRDRW, REELNU, FTRCNO

NAME = 1
ITB
IDSRCE, INREC
MOFF(1), MREC(1)
MOFF(2), MREC(2)                          Depending on the value of
 ...   ,  ...                                      INREC
MOFF(INREC), MREC(INREC)

NAME = 2
NTRCED, NAUTED
ICDPED(1), ITRCED(1), ...., ICDPED(NTRCED), ITRCED(NTRCED)
NEDST, NEDWIN, NEDLIM, EDLEV

NAME = 3
NTAP, NFIRST, NMUTE
MOFF(1), MUTE(1)
MOFF(2), MUTE(2)                          Depending on value of NFIRST
 ...   ,  ...                                      and NMUTE
MOFF(NMUTE), MUTE(NMUTE)

NAME = 4
NTRACP
ITRACP(1), ITRACP(2), ..., ITRACP(NTRACP)

NAME = 5

NAME = 6
ITRTYP, ITRAPL, FACT

NAME = 7
ITVAPL

NAME = 8
NWIN, IBOT

NAME = 9
FL, FU
FTPR1, FTPR2

NAME = 10
FLR, FUR

NAME = 11
IFFLG, NTOT, NLENG
NBEG                                      Depending on value of IFFLG
NBEG(1),NBEG(2),...,NBEG(NCHAN)
IWHITE
```

```
NAME = 12
IFFLG, NTOT, NLENG, NLAG
NBEG                                        Depending on value of IFFLG
NBEG(1),NBEG(2),...,NBEG(NCHAN)
IWHITE

NAME = 13
AUTFLG, NAUTOT
NAUBEG                                          Depending on value of
NAUBEG(1),NAUBEG(2),...,NAUBEG(NCHAN)                          AUTFLG

NAME = 14
NRMFLG

NAME = 15

NAME = 16
ISTK, INTERP, INTSW
NGATHR, (XSTART(I),I=1,NGATHR), XSTEP   Dependng on value of ISTK

NAME = 17
R(1), R(2), R(3)

NAME = 18
NOP, IVELSH, INTPVF, INTSVF
NGATHR, (XSTART(I),I=1,NGATHR), XSTEP

NAME = 19
IOP, XCMP, VAP

NAME = 20
NTIM1, FL1, FH1
NTIM2, FL2, FH2

NAME = 21
NHSUM

NAME = 22
IPSTR, IPEND
ICSTR, ICLEN, MAXSH, IDOC

NAME = 25

ILINE, ((ITEXT(I,J),I=1,64),J=1,ILINE)
```

The layout of the processing program parameter file is shown above with all processing options included. In practice, only some of the options would be selected. Mandatory entries in the data file are underlined.

Processing option entries are described in detail in subsections which now follow.

MANDATORY ENTRIES IN THE PROGRAM DATA FILE

NFILES (I6) — Number of input files to process. The program will terminate after this number of files, unless previously terminated from the keyboard at a tape change. A file is defined as a complete input CMP gather.

NCHAN (I6) — Numberof traces in each input gather. For pre-stack processing, NCHAN equals the total number of input traces including both seismic and auxillary channels. For post-stack processing, NCHAN is set to 1.

NSAMP (I6) — Number of samples in each input trace.

FSAMP (F10.0) — The sampling frequency of the input data in kilohertz (kHz).

IFIRST (I6) — The trace sequence number within input CMP gather of the first trace to include in processing.

ILAST (I6) — The trace sequence number within input CMP gather of the last trace to include in processing.

The above two parameters, IFIRST and ILAST, enable a subset of the traces in each input CMP gather to be selected.

ISTART (I6) — Specifies the recording delay on the output traces. If ISTART is positive or zero it is interpreted as the absolute time from the shot instant, in milliseconds, at which all the output traces are to start. If ISTART is negative, each output trace has the same delay as the input trace or CMP gather from which it originates.

ILEN (I6) — Length of output traces in milliseconds.

The above two parameters, ISTART and ILEN, specify a time window for the output seismic data trace. The length of the trace is defined by ILEN and may not vary along a given profile. The absolute time from shot instant to the start of the trace may change between CMPs or may be fixed to a specified value. The program will truncate or pad with zeros automatically at both the start and end of the input trace prior to any other processing as determined by these ouput parameters.

TPDRR (I6) — Input tape drive number (0, 1 or 2).

TPDRW (I6) — Output tape drive number (0, 1 or 2).

REELNU (I6) — The current output reel sequence number within profile.

FTRCNO (F10.0) — The current output trace sequence number within profile. The number is entered as a real variable to overcome the 16 bit limitation of the PDP 11 computer.

The following two parameters enable the user to specify text which is added to the ASCII tape header block output by the processing program. The text is entered at the first free space in the ASCII reel header that is greater that two lines long after line 25. The current date is automatically appended by the program for ease of referance.

ILINE (I6) — The number of lines of text to be added to output tape header block. (Maximum 10 lines).

If ILINE = 0, no text is added.

((ITEXT(I,J),I=1,64),J=1,ILINE) (64A1)

Lines of ASCII text describing processing. Each line can be up to 64 characters long.

If ILINE = 0 entries for ITEXT need not be provided.

5.1 TIME-BREAK AND SOURCE RECEIVER DEPTH CORRECTIONS

MAY ONLY BE CALLED ONCE PER PROCESSING RUN

NAME = 1

ITB (I6)	"Time-break" - the delay in milliseconds between the start of recording and the shot instant.
IDSRCE (I6)	= -1 Take source depth from trace header, see section 6. = 0 No source depth correction required. > 0 Give depth of source in meters (+ve)
INREC (I6)	= -1 Take receiver depth from trace headers, see section 6. = 0 No receiver depth correction required. > 0 Number of entries in receiver depth function table (maximum of 4)
MOFF(I), MREC(I) ... , ... (I6,I6)	The receiver depth function table, entered as: Receiver group number as used in the shot gathers, with receiver group number 1 nearest the ship, depth in metres to the specified group, (+ve). The program will linearly interpolate between points entered in the table. If INREC is set to -1 or 0, no receiver depth function table is required.

5.2 TRACE EDIT AND AUTOMATIC SPIKE DETECTION EDITING

MAY ONLY BE CALLED ONCE PER PROCESSING RUN

NAME = 2

NTRCED (I6)	The total number of bad input traces in the line being processed that are to be zeroed (maximum of 100). Excluding input traces that are already "dead" (ITYPE >2 in input trace header blocks). NTRCED must be updated if the processing program is stopped and then re-started. If explicit trace editing is not required, NTRCED = 0.

NAUTED (I6) Selects automatic noise spike detection editing:-
= 0 No spike edit.
= 1 Automatic spike edit.

If NTRCED>0 then:-

ICDPED(1), ITRCED(1), ..., ICDPED(NTRCED), ITRCED(NTRCED) (1215)

Where ICDPED(I),ITRCED(I) specify the traces to be zeroed as CMP gather number and trace number within the processed gather pairs.

These values need updating if the processing program is stopped and then re-started.

If NAUTED = 1 then:-

NEDST (I6) Absolute time in milliseconds to start of window to be scanned for noise spikes.

NEDWIN (I6) Length of spike scan window in milliseconds.

NEDLIM (I6) > 0 Spike width threshold in milliseconds. If any spikes exceed this limit then the whole trace is zeroed.
= -1 Only the samples that exceed the spike treshold, EDLEV, are zeroed.

EDLEV (F10.0) Spike amplitude theshold.

5.3 MUTE APPLICATION

NAME = 3

NTAP (I6)	Width in milliseconds of the cosine taper which is applied at the edge of the mute. If NTAP is positive the mute is applied to the start of the trace down to the time defined by the mute function. If NTAP is negative the mute is applied to the end of the trace from the time defined by the mute function.
NFIRST (I6)	The first trace in processed CMP gather to be muted. If NFIRST is negative the program will assume stacked data and automatically mute to the sea-floor, see section 4.3.
NMUTE (I6)	The number of entries in mute function table (maximum of 4).
MOFF(I), MUTE(I) ... , ... (I6,I6)	The mute function table, entered as: Processed CMP trace sequence number and the for I=1,NMUTE absolute time in milliseconds to the line of the mute function on that trace. The program will linearly interpolate between points entered in the table. The table is not required if sea-floor muting on stacked data is chosen.

5.4 POLARITY REVERSAL

MAY ONLY BE CALLED ONCE PER PROCESSING RUN

NAME = 4

NTRACP (I6)	Number of traces per processed CMP gather to be reversed.
ITRACP(1),..., FTRACP(NTRACP) (12I6)	Trace numbers within each processed CMP gather of traces to be reversed. The trace numbers can be entered in any order, up to 12 values per line.

5.5 SEA-BED CONSISTANT STATICS

MAY ONLY BE APPLIED TO UNSTACKED DATA

NAME = 5

(No other parameters in program data file; but depth values in "DK6:DEPTH.DAT" must be present - see section 4.3).

5.6 TIME RAMPS

NAME = 6

ITRTYP (I6) Specifies type of ramp:-

= 0 exp (FACT*T)

= 1 T exp (FACT*T)

= 2 T^2 exp (FACT*T)

= 3 T^2

Where T is absolute time from shot instant in seconds.

ITRAPL (I6) Specifies application or removal of ramp:-

= 0 removal
= 1 application

FACT (F10.0) Exponent scaling value in above expressions.

5.7 GEOMETRIC SPREADING CORRECTION

MAY ONLY BE APPLIED TO STACKED DATA

NAME = 7

ITVAPL (I6) Specifies application or removal of spreading correction:-

= 0 removal
= 1 application

5.8 AGC (Automatic Gain Control)

NAME = 8

NWIN	(I6)	Length in millisecondss of AGC averaging window.
IBOT	(I6)	= -1sea-bed following; AGC applied only to the portion of trace below the sea-bed. See section 4.3.
		= 0 AGC applied to whole trace.
		> 0 absolute time in milliseconds to the start of the AGC function, all samples in the trace that occur before given time have the same gain applied as that used at the given time.

5.9 BAND-PASS FILTER

Name = 9

FL	(F10.0)	Low cut frequency in kHz. i.e. the frequency at which the low-cut taper begins to reduce the gain from unity.
FU	(F10.0)	High cut frequency in kHz. i.e. the frequency at which the high-cut taper begins to reduce the gain from unity.
FTPR1	(F10.0)	Low-cut taper length in kHz. To avoid the introduction of "ringing", ideally FTPR1 > FL/2.
FTPR2	(F10.0)	High-cut taper length in kHz. To avoid the introduction of "ringing", ideally FTPR2 > FU,

5.10 BAND-REJECT FILTER

NAME = 10

FLR (F10.0) Frequency in kHz at which the taper on the low side of the rejection frequency begins to reduce the gain from unity.

FUR (F10.0) Frequency in kHz at which the taper on the high side of the rejection frequency begins to reduce the gain from unity.

The frequency rejected is the mid-point of FLR and FUR

$$F_{rej} = (FLR + FUR) / 2$$

5.11 WEINER SPIKING FILTER

NAME = 11

IFFLG (I6) Autocorrelation design flag. Takes values 0, 1, 2 or 3 and selects one of four possible methods of determining which section of trace is used to calculate the autocorrelation function needed for each filter design. See section 5.12.1 for details.

NTOT (I6) Length, in milliseconds, of the section of trace used to design the autocorrelation function.

NLENG (I6) Length of spiking filter in milliseconds.

NBEG (I6)
NBEG (1-NCHAN) (12I6)) One of these lines depending on the value of IFFLG.

Start time(s), in milliseconds, of the section of trace(s) used to calculate the autocorrelation function(s). See section 5.12.1 for details. If NBEG is less than the delay on the input trace then the program will automatically start the autocorrelation function at the start of the trace.

IWHITE (I6) % of pre-whitening to be applied. Used to limit noise amplification during the filtering process.

5.12 PREDICTION ERROR FILTER

NAME = 12

IFFLG (I6) — Autocorrelation design flag. Takes values 0, 1, 2, or 3 and selects one of four possible methods of determining which section of trace is used to calculate the autocorrelation function needed for each filter design. See section 5.12.1 for details.

NTOT (I6) — Length, in milliseconds, of the section of trace used to design the autocorrelation function.

NLENG (I6) — Length of prediction filter, in milliseconds.

NLAG (I6) — Prediction delay in milliseconds. For the special case of the prediction delay following the sea-bed, make NLAG negative. See section 2.12.4 for details.

NBEG (I6)
NBEG (1-NCHAN) (12I6)) — One of these lines, depending on the value of IFFLG.

Start time(s), is milliseconds, of the section of trace used to calculate the autocorrelation function(s). See section 5.12.1 for details.

IWHITE (I6) — % of pre-whitening to be applied. Used to limit noise amplification during the filter process.

5.12.1 Autocorrelation Design Options

See section 2.12.2 for more details.

IFFLG = 0 — One autocorrelation function is calculated for each CMP gather processed. A section from the first non-blank trace of the gather is used to calculate the autocorrelation function. The length of this section of trace is NTOT and the start is at absolute time NBEG in milliseconds. The same autocorrelation function is used to design the filters for all the traces of the CMP gather. Only one value of NBEG is required in the data file.

= 1 — A new autocorrelation function is calculated for each trace processed. The same section of each trace is used to calculate the autocorrelation functions. The sections of traces are NTOT long and begin at absolute time NBEG in milliseconds. Only one value of NBEG is required in the data file.

= 2 — A new autocorrelation function is calculated for each trace processed. The sections of trace used to calculate the autocorrelation functions follow the pattern of a mute previously applied to the trace. The length of the autocorrelation is NTOT milliseconds and begin at a delay of NBEG milliseconds from the end of the mute function. (Note that in this case NBEG is not an absolute time but a relative delay). Only one value of NBEG is required in the data file.

= 3 — A new autocorrelation function is calculated for each trace processed. The sections of traces used to calculate the autocorrelation functions are all NTOT long, but their positions along the traces are individually specified by the user. A value for the absolute start time of the trace section (in milliseconds) is entered for every trace of the CMP gather.
This option may only be used for CMP gathers of 96 seismic data traces or less.

5.13 AUTOCORRELATION FUNCTION OUTPUT TO DISC FILE

THIS MAY BE CALLED UPTO TWICE PER PROCESSING RUN

NAME = 13

AUTFLG (I6) — Autocorrelation design flag. Takes values 1, 2 or 3 and selects one of four possible ways of determining the section of a trace which is to give rise to the output autocorrelation function. Equivalent to IFFLG in the prediction error filter, see section 5.12.1.

NAUTOT (I6) — Length, in milliseconds, of the section of trace which is to give rise to the output autocorrelation function.
As the autocorrelation function is written to disc, the sample count NAUTOT * FSAMP, is rounded up to the nearest multiple of 128.

NAUBET (I6)
NAUBEG (1-NCHAN) (12I6)) — One of these lines depending on the value of AUTFLG.

Start time(s), in milliseconds, of the section of the trace(s) which give rise to the ouput auto-correlation function(s).

Note: The parameters AUTFLG, NAUTOT and NAUBEG above, are exactly analogous to the parameters IFFLG, NTOT and NBEG, respectively, in the specifications for the Weiner Filter and the Prediction Error Filter.

5.14 TRACE NORMALIZATION

NAME = 14

NRMFLG (I6) = -1 Normalize each trace so that the maximum amplitude is 1.
= 0 Normalize each trace to Unit RMS Energy.
= 1 Normalize each trace to the mean squared energy.

5.15 REFLECTION STRENGTH TRANSFORM (HILBERT)

NAME = 15

There are no more parameters used with this option.

5.16 NORMAL MOVE-OUT CORRECTION AND STACK

NAME = 16

ISTK (I6)
- = -2 Remove NMO correction from trace within processed CMP gather only.
- = -1 Apply NMO correction to trace within processed CMP gather only.
- = 1 Apply NMO correction to trace within processed CMP gather and stack to form single output trace for each CMP gather read.
- = 2 Stack previously NMO corrected traces within each processed CMP gather.

INTERP (I6) Level of sample interpolation to be used prior to applying or removing NMO correction to a given input trace. Maximum of 4.

INTSW (I6) Determines whether lateral interpolation between given velocity analysis points is required.
- = 0 for no lateral interpolation.
- = 1 for lateral interpolation.

See section 2.16.2.

NGATHR (I6) The number of different source/near-receiver offsets that appear cyclically on processed CMP gathers. Used for the case where the receiver spacing is less than or equal to the shot spacing. Maximum of 6.

XSTART (1-NGATHR) (F10.0)
The source/near receiver spacing(s) in metres.

XSTEP (F10.0) The receiver spacing in each processed CMP gather in metres.

Note:
If the Velocity Filter routine, see section 2.18, has been called prior to this in the same processing run; NGATHR, XSTART and XSTEP must be the same.

Special Option:
If, while demultiplexing or CMP sorting the data, the SEG-Y trace header has had the source/receiver word filled in, it is safer to use this entry to ensure the correct NMO is applied. In this case set NGATHR = 1, XSTART(1) = -1.0, and XSTEP = 0.0.

5.17 THREE TRACE MIX

MAY ONLY BE USED ON STACKED DATA

NAME = 17

R(1), R(2), R(3) (3F10.0) — The weighting ratios of the three traces to be mixed.

5.18 VELOCITY FILTERING

MAY ONLY BE USED ON UN-STACKED GATHERS

NAME = 18

NOP (I6) — The operator size, this determines the efficiency of the filter to remove unwanted coherent arrivals in the gather.

IVELSH (I6) — The pecentage shift to be applied to the RMS velocity/time data read from velocity filter data file DK6:VELFIL.DAT. If the number is less than or equal to 100% then coherent arrivals with apparent stacking velocities lower than the cut-off velocity will be suppressed, if IVELSH is greater than 100% then arrivals with apparent velocities greater the cut-off velocity will be removed.

INTPVF (I6) — Level of sample interpolation to be used prior to applying or removing NMO corrections to a given input trace. Maximum of 4.

INTSVF (I6) — Determines whether lateral interpolation between given velocity analysis points is required.
= 0 for no lateral interpolation.
= 1 for lateral interpolation.
See section 2.16.2.

NGATHR (I6), XSTART (1-NGATHR) (F10.0), XSTEP (F10.0)
As in NMO and stack routine above, see section 2.16.

Note:
If a Normal Move-out routine, see section 2.16, has been called prior to this in the same processing run; NGATHR, XSTART and XSTEP must be the same.

5.19 DIP FILTER

MAY ONLY BE USED ON STACKED DATA

NAME = 19

IOP	(I6)	The dimension of the 2-D convolutional operator for the dip filter. This determines the efficiency of the filter.
XCMP	(F10.0)	The horizontal spacing of the CMP gathers in metres.
VAP	(F10.0)	The minimum apparent velocity that will be passed by the filter, km/sec.. All apparent coherent arrivals with velocity less than VAP/2 will be totally rejected.

5.20 TIME VARYING BAND-PASS FILTER

NAME = 20

NTIM1	(I6)	The absolute time in milliseconds upto which the filter parameters are described by FL1 and FH1. If negative, represents delay after sea-floor, see section 4.3.
FL1	(F10.0)	The lower cut-off frequency of band-pass filter at time NTIM1, kHz.
FH1	(F10.0)	The upper cut-off frequency of band-pass filter at time NTIM1, kHz.
NTIM2	(I6)	The absolute time in milliseconds after which the filter parameters are described by FL2 and FH2. If negative, repesents delay after sea-floor.
FL2	(F10.0)	The lower cut-off frequency of band-pass filter at time NTIM2, kHz.
FH2	(F10.0)	The upper cut-off frequency of band-pass filter at time NTIM2, kHz.

5.21 HORIZONTAL SUMNATION OF STACKED DATA

MAY ONLY BE USED ON STACKED DATA

NAME = 21

NHSUM (I6) The number of input traces to sum for each output trace. The total number of stacked traces is reduced by this factor but the effective fold of cover is increased and the CMP spacing is also increased by factor.
N.B. If subsequently dip filtering the data give the new CMP spacing resulting from this process.

5.22 AUTOMATIC CMP CONSISTANT STATICS

MAY ONLY BE USED ON UNSTACKED DATA

NAME = 22

IPSTR (I6) First trace within processed gather to use in pilot trace.

IPEND (I6) Last trace within processed gather to use in pilot trace.

IXSTR (I6) Absolute start time of the cross correlation window between each processed trace and the brute stacked pilot trace, milliseconds.

IXLEN (I6) Length of the cross correlation window, milliseconds.

MAXSH (I6) Maximum trace shift allowed in either direction, milliseconds.

IDOC (I6) When processing gathers of less than 24 fold it is possible to have the total autostatic correction applied listed in the log file. This is only advised for short sections of data as the log file may become very long.

= 0 No listing in log file.

= 1 Include listing in log file.

The indicates in processed trace order the number of samples each has been moved. +ve indicate the trace has been moved up, -ve that the trace has been moved down with respect to the pilot trace.

6. DURHAM SEG-Y REEL AND TRACE HEADER DESCRIPTION

Reel Headers

There are two reel header blocks required by the SEG-Y standard. The first is in a text format than can be directly read to establish the nature of the data on the given tape. The second is a binary block that contains the reel parameters in a format the computer may read. In the case of DURHAM SEG-Y the text block is written in ASCII code as EBCDIC is not supported by the PDP-11/34 computer under the RT-11 operating system. It is assumed that a proceesing run will start with the input tape at its load point, i.e. at the beginning, in which case both header blocks are read and stored in a disc file DK6:SEISHD.DAT, the tape headers from all subsquent input tapes mounted during the given run will be ignored. If the processing run starts part way through an input tape, the program assumes a re-start and will use the current version of DK6:SEISHD.DAT on disc. Therefore, it is strongly advised that if starting a new processing run part way through an input tape that a dummy run is made with the file counter set to 1 and the input tape at its load point. This ensures that the correct version of the reel header blocks are stored in the disc file.

The following entries in the ASCII tape header block are updated before writing the tape headers to a new output tape :

REEL NO
DATA TRACES/RECORD
SAMPLES/TRACE

Also, any text entered is added in the next available space in the the ASCII header, along with the current date. Any other entries in the input ASCII header are passed unaltered to the output ASCII header.

The entries in the binary tape header block that are updated before writing the tape headers to a new output tape are:

REEL NO.	(bytes 9-12)
NO. OF DATA TRACES/RECORD	(bytes 13-14)
NO. OF SAMPLES PER TRACE	(bytes 21-22)

Any other entries in the input binary header are passed unaltered to the output binary header.

Trace Headers

Input Data

The trace header of every trace input to the program is interrogated and the following entries must be present:

CMP GATHER NO.	(bytes 21-24)	JCDPNO
TRACE NO. WITHIN CMP GATHER	(bytes 25-28)	JCDPTR
TRACE IDENTIFICATION CODE	(bytes 29-30)	ITYPE
RECORDING DELAY (IN MILLISEC)	(bytes 109-110)	IDELAY
NO. OF SAMPLES PER TRACE	(bytes 115-116)	ISAMP
SAMPLE INTERVAL (IN uSEC)	(bytes 117-118)	ITIME

Additionally, if XSTART in the STACK program data file is negative, there must be an entry for:

SOURCE/RECEIVER OFFSET (IN METRES)	(bytes 37-40)	JOFF

If source receiver statics are to be applied then the following entries may be included:

RECEIVER GROUP ELEVATION (IN METRES)	(bytes 41-44)	JDREC
SOURCE DEPTH BELOW SURFACE (IN METRES)	(bytes 49-52)	JDSOR

N.B. The receiver number is negative when below surface, where as the source number is positive. A velocity of 1.5 km/sec is assumed, i.e. water velocity.

Data processed with the Durham suite of programs will automatically have all the mandatory trace header entries present.

Output Data

The program constructs trace header blocks for the output stacked traces by setting values for the following parameters:

CMP GATHER NO.	(bytes 21-24)	JCDPNO
TRACE NO WITHIN CMP GATHER	(bytes 25-28)	JCDPTR

The CMP number is passed unaltered to the output trace, the trace number is modified such that each CMP starts with trace 1 and increments to the last trace; which in the case of un-stacked data equals the number of traces selected from the input gather; and for stacked data is always set to 1.

TRACE IDENTITY CODE (bytes 29-30) ITYPE

This parameter is passed unchanged unless the given trace has been edited, in which case it is set to 2, "dead trace". In a stacked trace header, this parameter is set to 1, unless every trace involved in the stack was blank, in which case it is set to 2.

NO. OF VERTICALLY STACKED TRACES (bytes 31-32) IFOLD

Passed without modification unless a stacking option has been used in the processing run when this parameter is set equal to the number of non-blank input traces that have contributed to the stacked trace.

NO. OF HORIZONTALLY SUMMED TRACES (bytes 33-34) NVSUM

This entry represents the number of input traces summed for each output trace when horizontally summing stacked traces, see section 2.21.

SOURCE/RECEIVER OFFSET (bytes 37-40) JOFF

This parameter is passed unchanged for un-stacked data or set to 0 in stacked data.

RECORDING DELAY (IN MILLISEC)	(bytes 109-110)	IDELAY
NO. OF SAMPLES PER TRACE	(bytes 115-116)	ISAMP
SAMPLE INTERVAL (IN uSEC)	(bytes 117-118)	ITIME

The recording delay and number of output samples are modified according to the output trace data entered in the parameter file. The sample interval is passed unchanged.

Also, if a mute has been applied to data input to the program, details will be supplied in the trace header entries:

ABSOLUTE TIME TO START OF MUTED SECTION OF TRACE (IN MILLISEC)	(bytes 111-112)	MUTST
ABSOLUTE TIME TO END OF MUTED SECTION OF TRACE (IN MILLISEC)	(bytes 113-114)	MUTEND

If any static corrections have been applied to unstacked data then the following entries will also be written:

SOURCE STATIC CORRECTION (IN MTERES)	(bytes 99-100)	ISST
RECEIVER STATIC CORRECTION (IN METRES)	(bytes 101-102)	IRST
TOTAL STATIC CORRECTION (IN MILLISEC)	(bytes 103-104)	ITOTST

In addition, in the "unassigned" area of a stacked trace header, details of the velocity profile used to stack the trace are recorded:

NUMBER OF INTERFACES IN THE VELOCITY PROFILE USED TO STACK THE TRACE	(bytes 199-200)	ILYR
TWO-WAY TRAVEL TIMES IN MILLISECS ONE VALUE FOR EACH INTERFACE; UP TO 10 VALUES	(bytes 201-220) (2 bytes/value)	IT0LYR
R.M.S. STACKING VELOCITY IN METRES/SEC (NOT KM/SEC as in program data file) ONE VALUE FOR EACH INTERFACE; UP TO 10 VALUES	(bytes 221-240) (2 bytes/value)	IVLYR
TRACE SEQUENCE NUMBER WITHIN LINE	(bytes 1-4)	JTRCNO

This is the absolute output trace number within line. The first value is defined in the parameter file and it is incremented for each trace written to tape regardless of CMP gather or tape number.

All other parts of a stacked trace header not mentioned here are blanked or passed unchanged.

For the autocorrelation files; DR6:AUTO1.DAT and DR6:AUTO2.DAT; only trace header from the first trace processed is saved and written to the disc file into the first disc block. Most of the arguments are passed un-changed accept for:

TRACE IDENTIFICATION CODE, bytes 29-30, is set to 9.

NO. OF SAMPLES/TRACE, bytes 115-116, is set to the number of samples in the autocorrelation function.

NO. OF TRACES/PROCESSED GATHER, bytes 35-36, is set to the number of traces in processed gather for use by the PLOT program.

```
C
C  THIS IS AN INTERACTIVE PROGRAM TO VERIFY
C  DATA FILES FOR THE RUNNING OF THE SEISMIC
C  PROCESSING PACKAGE
C
C  WRITTEN R.W.HOBBS SEP 1984
C
      VIRTUAL ICDPED(100),ITRCED(100),ITRACP(48),ITEXT(64,10)
      REAL*4 R(3),XSTART(6),SEIS(2)
      INTEGER*2 REELNU,TPDRR,TPDRW,NBEG(96),NAUBEG(96),
     +AUTFLG,MOFF(4),MTIM(4),MREC(4)
      LOGICAL*1 FNAME(16),IANS,LDUM,ITEXT
C
      DATA SEIS(1)/6RDK1SEI/,SEIS(2)/6RS  SAV/
C
      IPFLG=0
      TYPE 4000
 4000 FORMAT(' INTERACTIVE INPUT PROGRAM FOR THE SEISMIC',
     +' PROCESSING PACKAGE',/,' ALSO DATA FILE VERIFICATION',//,
     +' Give full name of data file containing processing',
     +' parameters',/,'$<CR> for interactive input ? ')
      ACCEPT 4100,FNAME
 4100 FORMAT(16A1)
      IF(FNAME(1).NE.'D') GOTO 7
      IPFLG=1
C
C  OPEN DATA FILE
C
      OPEN(UNIT=1,TYPE='OLD',NAME=FNAME,ERR=6)
      GOTO 7
 6    STOP 'INPUT DATA FILE NOT FOUND'
C
C  OPEN NEW DATA FILE FOR CHECKED DATA
C
 7    TYPE 4200
 4200 FORMAT(/,' THE OUTPUT DATA FILE NAME IS DK6:SEIS.DAT')
      OPEN(UNIT=2,TYPE='NEW',NAME='DK6:SEIS.DAT',ERR=8)
      GOTO 9
 8    STOP 'CANNOT OPEN OUTPUT DATA FILE DK6:SEIS.DAT'
C
C  READ I/P FILE LINE BY LINE, CHECK AND O/P TO DK6:SEIS.DAT
C
 9    IF(IPFLG.EQ.0) GOTO 415
      READ(1,1005)NFILES,NCHAN,NSMPIN,FSAMP
 1000 FORMAT(12I6)
 415  TYPE 6000
 6000 FORMAT(/,' INPUT PARAMETERS',//,
     +'$Give number of files to process (integer) ? ')
      ACCEPT 1000,NFILES
      IF(IPFLG.EQ.1) GOTO 416
      TYPE 6001
 6001 FORMAT('$Give number of traces per file (integer) ? ')
      ACCEPT 1000,NCHAN
      TYPE 6002
 6002 FORMAT('$Give number of samples per input trace (integer) ? ')
      ACCEPT 1000,NSMPIN
      TYPE 6003
 6003 FORMAT('$Give sampling frequency kHz (real) ? ')
      ACCEPT 1100,FSAMP
 416  WRITE(2,5005)NFILES,NCHAN,NSMPIN,FSAMP
 5005 FORMAT(3(I5,','),F9.4,',')
      IF(IPFLG.EQ.0) GOTO 420
      READ(1,1000)IFIRST,ILAST,ISTART,ILEN
      GOTO 425
 420  TYPE 6004
 6004 FORMAT(/,' OUTPUT PARAMETERS',//,
     +' Give first and last trace number(s) in input gather to include',
     +' in processing',/,'$<CR> for all (integer) ?,? ')
      ACCEPT 1000,IFIRST,ILAST
      IF(IFIRST.EQ.0) IFIRST=1
      IF(ILAST.EQ.0) ILAST=NCHAN
      TYPE 6005
```

```
 6005 FORMAT('$Give start time of output trace in msec',
     +' -1 for same as input (integer) ? ')
      ACCEPT 1000,ISTART
      TYPE 6006
 6006 FORMAT('$Give length of output trace in msec',
     +' -1 for same as input (integer) ? ')
      ACCEPT 1000,ILEN
 425  IF(ILEN.EQ.-1) ILEN=IFIX(FLOAT(NSMPIN)/FSAMP)
      WRITE(2,5000)IFIRST,ILAST,ISTART,ILEN
 5000 FORMAT(12(I5,','))
 1005 FORMAT(3I6,F10.0)
 1100 FORMAT(3F10.0)
      IF(IPFLG.EQ.0) GOTO 430
      READ(1,1005)TPDRR,TPDRW,REELNU,FTRCNO
 430  TYPE 6007
 6007 FORMAT(/,' TAPE I/P AND O/P PARAMETERS',//,
     +'$Give input and output tape drive numbers (integer) ?,? ')
      ACCEPT 1000,TPDRR,TPDRW
C
C  CHECK REEL CONSTANTS
C
 435  TYPE 3400
 3400 FORMAT('$Give current output reel sequence no. (integer) ? ')
      ACCEPT 1000,REELNU
      IF(REELNU.EQ.0) REELNU=1
      TYPE 3500
 3500 FORMAT('$Give next trace output sequence no. (integer) ? ')
      ACCEPT 1100,FTRCNO
      IF(FTRCNO.EQ.0.0) FTRCNO=1.0
      WRITE(2,3505)TPDRR,TPDRW,REELNU,FTRCNO
 3505 FORMAT(3(I5,','),F9.0,',')
C
C  SET UP CONSTANTS
C
      NTRACO=ILAST-IFIRST+1
      NTRACI=NTRACO
      NSAMP=IFIX(FLOAT(ILEN)*FSAMP)
      NSAMP2=2
 1    IF(NSAMP2.GE.NSAMP)GOTO 2
      NSAMP2=NSAMP2*2
      GOTO 1
 2    IP=0
      IST=0
      IVF=0
      NFUNC=0
      NAUTED=0
      IFILT=0
      IDECON=0
      IAUTO=0
      IDFLG=0
      NOFF=0
      NGATHR=1
      XSTART(1)=-1.0
      XSTEP=0.0
      IA=0
      IB=IA+NSAMP
      IC=IB+NSAMP
      ID=IC+NSAMP
      IG=24576-NSAMP
      IE=IG-NSAMP
      IF(4*NSAMP.GT.IE.OR.2*NSAMP2+2.GT.IE) STOP'O/P TRACES TOO LONG'
C
C READ IN JOB SPECIFIC DATA AND SET UP
C THE FILTERS TO BE USED
C
      IF(IPFLG.EQ.1) TYPE 2000
 2000 FORMAT(/,' PROCESSING SUMMARY',/)
 5    IP=IP+1
      IF(IP.GT.20) STOP'TOO MANY OPTIONS'
      IF(IPFLG.EQ.1) READ(1,1000) NAME
      IF(IPFLG.EQ.0) TYPE 6008
 6008 FORMAT(/,' PROCESSING OPTIONS',
```

```
     + /,'  1     Time break, source and receiver depth correction',
     + /,'  2     Trace edit',
     + /,'  3     Mute at start and/or end of trace',
     + /,'  4     Polarity reversal',
     + /,'  5     Sea-floor statics on un-stacked data',
     + /,'  6     Time varying gain funcions to apply or remove',
     + /,'  7     (TV)**2 gain function to apply or remove',
     + /,'  8     Automatic gain control',
     + /,'  9     Band-pass filter',
     + /,' 10     Band-reject filter',
     + /,' 11     Wiener spiking filter',
     + /,' 12     Prediction error filter',
     + /,' 13     Auto correlation output to DR6:',
     + /,' 14     Trace normalization',
     + /,' 15     Hilbert transform to reflection strength',
     + /,' 16     Normal move-out correction to apply or remove and',
     +           '/or stack',
     + /,' 17     Horizontal three trace mix on stacked data',
     + /,' 18     Velocity filter un-stacked gathers',
     + /,' 19     Dip filter stacked data',
     + /,' 20     Time varying band-pass filter',
     + /,' 21     Horizontal sum of stacked data',
     + /,' 22     Auto statics on NMOed unstacked gathers',
     + /,' 25     Exit and output trace to tape',
     + /,'$ GIVE OPTION NUMBER ? ')
      IF(IPFLG.EQ.0) ACCEPT 1000,NAME
      IF(IPFLG.EQ.0) TYPE 6009
 6009 FORMAT(/)
      WRITE(2,5000) NAME
      GOTO (10,20,30,40,50,60,70,80,90,100,110,
     +120,130,140,150,160,170,180,190,200,210,
     +220,250,250,250) NAME
C
C  TIME BREAK CORRECTION
C
 10   TYPE 2010
 2010 FORMAT(' TIME BREAK AND SOURCE/RECEIVER STATIC CORRECTION')
      IF(IPFLG.EQ.0) GOTO 11
      READ(1,1000) ITB
      READ(1,1000) IDSRCE,INREC
      IF(INREC.LE.0) GOTO 12
      DO 13 I=1,INREC
 13   READ(1,1000) MOFF(I),MREC(I)
      GOTO 12
 11   TYPE 6010
 6010 FORMAT('$Give time break correction msec (integer) ? ')
      ACCEPT 1000,ITB
      TYPE 6011
 6011 FORMAT('$Are source/receiver depths to be taken',
     +' from trace header (Y/N) ? ')
      ACCEPT 3000,IANS
      IDSRCE=-1
      INREC=-1
      IF(IANS.EQ.'Y') GOTO 12
      TYPE 6012
 6012 FORMAT('$Give source depth below datum m (integer) ? ')
      ACCEPT 1000,IDSRCE
      TYPE 6013
 6013 FORMAT(' Give receiver depth function as; trace number in',
     +' shot gather, depth m',/,' (integer, max. 4 pairs)')
      INREC=0
      DO 15 I=1,4
      TYPE 6035
      ACCEPT 1000,MOFF(I),MREC(I)
      IF(MOFF(I).EQ.0) GOTO 12
 15   INREC=INREC+1
 12   WRITE(2,5000) ITB
      WRITE(2,5000) IDSRCE,INREC
      IF(INREC.LE.0) GOTO 5
      DO 16 I=1,INREC
 16   WRITE(2,5000) MOFF(I),MREC(I)
      GOTO 5
```

```
C
C TRACE EDIT DATA
C
 20   TYPE 2020
 2020 FORMAT(' TRACE EDIT')
      IF(IPFLG.EQ.0) GOTO 21
      READ(1,1000) NTRCED,NAUTED
      GOTO 22
 21   TYPE 6020
 6020 FORMAT('$Give total number of traces to edit <100 (integer) ? ')
      ACCEPT 1000,NTRCED
      TYPE 6021
 6021 FORMAT('$Automatic editting of traces with spikes (Y/N) ? ')
      ACCEPT 3000,IANS
      IF(IANS.EQ.'Y') NAUTED=1
 22   IF(NTRCED.GT.100) STOP'TOO MANY TRACES TO EDIT'
      WRITE(2,5000) NTRCED,NAUTED
      IF(NTRCED.EQ.0) GOTO 25
      IF(IPFLG.EQ.0) TYPE 6022
 6022 FORMAT(' Give traces to edit in',
     +' CMP then trace order in output gather')
      DO 23 I=1,NTRCED
      IF(IPFLG.EQ.0) GOTO 24
      READ(1,1000) ICDPED(I),ITRCED(I)
      GOTO 23
 24   TYPE 6023
 6023 FORMAT('$CMP,Trace no. ?,? ')
      ACCEPT 1000,ICDPED(I),ITRCED(I)
 23   WRITE(2,5000) ICDPED(I),ITRCED(I)
 25   IF(NAUTED.EQ.0) GOTO 5
      IF(IPFLG.EQ.0) GOTO 27
      READ(1,1005) NEDST,NEDWIN,NEDLIM,EDLEV
      GOTO 29
 27   TYPE 6024
 6024 FORMAT('$Automatic edit window start and length msec',
     +' (integer) ?,? ')
      ACCEPT 1000,NEDST,NEDWIN
      TYPE 6025
 6025 FORMAT('$Give minimum absolute spike value (+ve real) ? ')
      ACCEPT 1100,EDLEV
      TYPE 6026
 6026 FORMAT(' Give maximum allowable number of successive spikes',
     +/,'$-1 if only flagged spikes are to be zeroed (integer) ? ')
      ACCEPT 1000,NEDLIM
 29   WRITE(2,6029) NEDST,NEDWIN,NEDLIM,EDLEV
 6029 FORMAT(3(I5,','),1PE9.3,',')
      GOTO 5
C
C MUTE DATA
C
 30   TYPE 2030
 2030 FORMAT(' MUTE')
      IF(IPFLG.EQ.0) GOTO 31
      READ(1,1000) NTAP,NFIRST,NFUNC
      IF(NFUNC.EQ.0.OR.NFIRST.LT.0) GOTO 32
      DO 33 I=1,NFUNC
 33   READ(1,1000) MOFF(I),MTIM(I)
      GOTO 32
 31   TYPE 6030
 6030 FORMAT('$Give taper length msec (integer) ? ')
      ACCEPT 1000,NTAP
      IF(NCHAN.GT.1) GOTO 34
      TYPE 6033
 6033 FORMAT('$For stacked data is mute to sea-floor required (Y/N) ? ')
      ACCEPT 3000,REP
      IF(REP.NE.'Y') GOTO 34
      NFIRST=-1
      IDFLG=IDFLG+1
      GOTO 32
 34   TYPE 6031
 6031 FORMAT('$Give number of first trace to mute in processed',
     +' gather ? ')
```

```
      ACCEPT 1000,NFIRST
      TYPE 6032
 6032 FORMAT('$Apply mute at (S)tart or (E)nd of trace (S/E) ? ')
      ACCEPT 3000,IANS
      IF(IANS.EQ.'E') NTAP=-NTAP
      TYPE 6034
 6034 FORMAT(' Give mute function as; trace number in processed gather,',
     +' mute time msec',/,' (integer, max. 4 pairs)')
      NFUNC=0
      DO 35 I=1,4
      TYPE 6035
 6035 FORMAT('$?,? ')
      ACCEPT 1000,MOFF(I),MTIM(I)
      IF(MOFF(I).EQ.0) GOTO 32
 35   NFUNC=NFUNC+1
 32   WRITE(2,5000) NTAP,NFIRST,NFUNC
      IF(NFUNC.EQ.0.OR.NFIRST.LT.0) GOTO 5
      DO 36 I=1,NFUNC
 36   WRITE(2,5000) MOFF(I),MTIM(I)
      GOTO 5
C
C POLARITY REVERSAL DATA
C
 40   TYPE 2040
 2040 FORMAT(' POLARITY REVERSAL')
      IF(IPFLG.EQ.0) GOTO 41
      READ(1,1000) NTRCPL
      IF(NTRCPL.GT.48) STOP'TOO MANY POLARITY REVERSALS'
      READ(1,1000)(ITRACP(I),I=1,NTRCPL)
      GOTO 42
 41   TYPE 6040
 6040 FORMAT('$Give number of traces to reverse in each gather <48',
     +' (integer) ? ')
      ACCEPT 1000,NTRCPL
      TYPE 6041,NTRCPL
 6041 FORMAT(' Give',I3,' trace number(s) in processed gather',
     +' (integer, 12 per line) ? ')
      ACCEPT 1000,(ITRACP(I),I=1,NTRCPL)
 42   WRITE(2,5000) NTRCPL
      WRITE(2,5000)(ITRACP(I),I=1,NTRCPL)
      GOTO 5
C
C  STATICS
C
 50   TYPE 2050
 2050 FORMAT(' SEA-FLOOR STATICS')
      IDFLG=IDFLG+1
      GOTO 5
C
C TIME RAMP DATA
C
 60   IF(IPFLG.EQ.0) GOTO 61
      READ(1,1300) ITRTYP,ITRAPL,FACT
 1300 FORMAT(2I6,F10.0)
      TYPE 2061
 2061 FORMAT('$TIME')
      IF(ITRAPL.GT.0) TYPE 2062
 2062 FORMAT('  RAMP APPLIED')
      IF(ITRAPL.EQ.0) TYPE 2063
 2063 FORMAT('  RAMP REMOVED')
      GOTO 62
 61   TYPE 6060
 6060 FORMAT(' TIME RAMP',/,
     +'  0      exp(ft)',/,
     +'  1      t*exp(ft)',/,
     +'  2      t*t*exp(ft)',/,
     +'  3      t*t',/,
     +'$        Give option number ? ')
      ACCEPT 1000,ITRTYP
      TYPE 6061
 6061 FORMAT('$Apply=1 or remove=0 ? ')
      ACCEPT 1000,ITRAPL
```

```
      IF(ITRTYP.EQ.3) GOTO 62
      TYPE 6062
 6062 FORMAT('$Give exponent factor, f (real) ? ')
      ACCEPT 1100,FACT
 62   WRITE(2,5300) ITRTYP,ITRAPL,FACT
 5300 FORMAT(2(I5,','),F9.4,',')
      GOTO 5
C
C TV**2 RAMP
C
 70   IF(IPFLG.EQ.0) GOTO 71
      READ(1,1000) ITVAPL
      TYPE 2071
 2071 FORMAT('$(TV)**2 GEOMETRIC SPREADING CORRECTION')
      IF(ITVAPL.GT.0) TYPE 2062
      IF(ITVAPL.EQ.0) TYPE 2063
      GOTO 72
 71   TYPE 6070
 6070 FORMAT(' (TV)**2 RAMP',/,
     +'$Apply=1 or remove=0 ? ')
      ACCEPT 1000,ITVAPL
 72   WRITE(2,5000) ITVAPL
      GOTO 5
C
C  AGC
C
 80   TYPE 2080
 2080 FORMAT('$AGC')
      TYPE 2062
      IF(IPFLG.EQ.0) GOTO 81
      READ(1,1000) NWIN,ISTART
      GOTO 82
 81   TYPE 6080
 6080 FORMAT('$Give AGC window length msec (integer) ? ')
      ACCEPT 1000,NWIN
      TYPE 6081
 6081 FORMAT(' AGC start options',/,
     +' -1      Start at sea-floor',/,
     +'  0      Start at beginning of trace',/,
     +' >0      Start at time specified msec',/,
     +'$        Give option (integer) ? ')
      ACCEPT 1000,ISTART
 82   NHWIN=NWIN/2
      IF(NHWIN.EQ.0) STOP'NO AGC WINDOW GIVEN'
      IF(ISTART.LT.0) GOTO 85
      IF(ISTART.GT.NHWIN) GOTO 87
      ISTART=NHWIN
      GOTO 87
 85   IDFLG=IDFLG+1
 87   WRITE(2,5000) NWIN,ISTART
      GOTO 5
C
C BANDPASS FILTER
C
 90   TYPE 2090
 2090 FORMAT('$BANDPASS')
      TYPE 2091
 2091 FORMAT('  FILTER')
      IF(IPFLG.EQ.0) GOTO 91
      READ(1,1100)FL,FU
      READ(1,1100)FTPR1,FTPR2
      GOTO 92
 91   TYPE 6090
 6090 FORMAT('$Give lower and upper pass-band limits kHz (real) ?,? ')
      ACCEPT 1100,FL,FU
      TYPE 6091
 6091 FORMAT('$Give lower and upper taper lengths kHz (real) ?,? ')
      ACCEPT 1100,FTPR1,FTPR2
 92   IF(FL.GT.FSAMP.OR.FU.GT.FSAMP
     +    .OR.FTPR1.GT.FSAMP.OR.FTPR2.GT.FSAMP) GOTO 93
      WRITE(2,5100)FL,FU
 5100 FORMAT(6(F9.4,','))
```

```
      WRITE(2,5100)FTPR1,FTPR2
      GOTO 5
 93   IF(IPFLG.NE.0) STOP'I/P FREQUENCY IN HZ'
      TYPE 6093
 6093 FORMAT(' ∘ WARNING ∘ FREQUENCY INPUT IN KHZ')
      GOTO 90
C
C BANDREJECT FILTER
C
 100  TYPE 2100
 2100 FORMAT('$BAND-REJECT')
      TYPE 2091
      IF(IPFLG.EQ.0) GOTO 101
      READ(1,1100)FLR,FUR
      GOTO 102
 101  TYPE 6100
 6100 FORMAT('$Give lower and upper limits of reject window kHz',
     +' (real) ?,? ')
      ACCEPT 1100,FLR,FUR
 102  IF(FLR.GT.FSAMP.OR.FUR.GT.FSAMP) GOTO 103
      WRITE(2,5100)FLR,FUR
      GOTO 5
 103  IF(IPFLG.NE.0) STOP'I/P FREQUENCY IN HZ'
      TYPE 6093
      GOTO 100
C
C WIENER AND PREDICTION ERROR FILTER
C
 110  TYPE 2110
 2110 FORMAT('$WIENER')
      GOTO 115
 120  TYPE 2120
 2120 FORMAT('$PREDICTION')
 115  TYPE 2122
 2122 FORMAT('  DECONVOLUTION')
      IFILT=IFILT+1
      IF(IFILT.GT.3) STOP'ONLY 3 DECONVOLUTION CALLS ALLOWED'
      IF(IPFLG.EQ.0) GOTO 122
      READ(1,1000) IFFLG,NTOT,NLENG,NLAG
      IF(IFFLG.LE.2) READ(1,1000) NBEG(1)
      IF(IFFLG.EQ.3) READ(1,1000) (NBEG(I),I=1,NTRACO)
      READ(1,1000) IWHITE
      GOTO 124
 122  TYPE 6120
 6120 FORMAT(' Autocorrelation design and start time opions',/,
     +'  0     First non-zero trace in gather only at a constant',
     +       ' start time',/,
     +'  1     For each trace in gather at a constant start time',/,
     +'  2     For each trace following previously applied',
     +       ' mute function',/,
     +'  3     For each trace following user specified time function',/,
     +'$       Give option number ? ')
      ACCEPT 1000,IFFLG
      TYPE 6121
 6121 FORMAT('$Give autocorrelation window length msec (integer) ? ')
      ACCEPT 1000,NTOT
      TYPE 6122
 6122 FORMAT('$Give total active filter length msec (integer) ? ')
      ACCEPT 1000,NLENG
      NLAG=0
      IF(NAME.EQ.11) GOTO 123
      IF(NCHAN.GT.1) GOTO 121
      REP=0
      TYPE 6128
 6128 FORMAT('$Is sea-floor consistant decon required on stacked data',
     +' (Y/N) ? ')
      ACCEPT 3000,REP
      IF(REP.EQ.'Y') TYPE 6129
 121  TYPE 6123
 6123 FORMAT('$Give prediction filter gap msec (integer) ? ')
 6129 FORMAT('$Give delay above sea-floor to  start of active filter',
     +' msec (integer) ? ')
```

```
      ACCEPT 1000,NLAG
      IF(REP.EQ.'Y') NLAG=-NLAG
 123  IF(IFFLG.LE.1) TYPE 6124
      IF(IFFLG.EQ.2) TYPE 6125
      IF(IFFLG.EQ.3) TYPE 6126,NTRACO
 6124 FORMAT('$Give autocorr. start time msec (integer) ? ')
 6125 FORMAT('$Give delay after mute function for autocorr.',
     +' start msec (integer) ? ')
 6126 FORMAT(' Give',I3,' start time(s) for autocorr. function',
     +' msec (integer, 12 per line) ? ')
      IF(IFFLG.LE.2) ACCEPT 1000,NBEG(1)
      IF(IFFLG.EQ.3) ACCEPT 1000,(NBEG(I),I=1,NTRACO)
      TYPE 6127
 6127 FORMAT('$Give pre-whitening percentage (integer) ? ')
      ACCEPT 1000,IWHITE
 124  IF(IFFLG.EQ.0) IDECON=IDECON+1
      IF(IDECON.GT.1) STOP'ONLY ONE TYPE 0 DECON ALLOWED'
      WRITE(2,5000) IFFLG,NTOT,NLENG,NLAG
      IF(IFFLG.LE.2) WRITE(2,5000) NBEG(1)
      IF(IFFLG.EQ.3) WRITE(2,5000) (NBEG(I),I=1,NTRACO)
      WRITE(2,5000) IWHITE
      IF(NAME.EQ.12.AND.NLAG.LE.0) IDFLG=IDFLG+1
      GOTO 5
C
C  AUTOCORRELATION OUTPUT
C
 130  TYPE 2130
 2130 FORMAT(' AUTOCORR. O/P TO DR6:')
      IAUTO=IAUTO+1
      IF(IAUTO.GT.2) STOP'ONLY 2 ALLOWED'
      IF(IPFLG.EQ.0) GOTO 131
      READ(1,1000) AUTFLG,NAUTOT
      IF(AUTFLG.LE.2) READ(1,1000) NAUBEG(1)
      IF(AUTFLG.EQ.3) READ(1,1000) (NAUBEG(I),I=1,NTRACO)
      GOTO 134
 131  TYPE 6130
 6130 FORMAT(' Autocorrelation design and start time opions',/,
     +'  1      For each trace in gather at a constant start time',/,
     +'  2      For each trace following previously applied',
     +       ' mute function',/,
     +'  3      For each trace following user specified time function',/,
     +'$        Give aptian number ? ')
      ACCEPT 1000,AUTFLG
      TYPE 6121
      ACCEPT 1000,NAUTOT
      IF(AUTFLG.LE.1) TYPE 6124
      IF(AUTFLG.EQ.2) TYPE 6125
      IF(AUTFLG.EQ.3) TYPE 6126,NTRACO
      IF(AUTFLG.LE.2) ACCEPT 1000,NAUBEG(1)
      IF(AUTFLG.EQ.3) ACCEPT 1000,(NAUBEG(I),I=1,NTRACO)
 134  WRITE(2,5000) AUTFLG,NAUTOT
      IF(AUTFLG.LE.2) WRITE(2,5000) NAUBEG(1)
      IF(AUTFLG.EQ.3) WRITE(2,5000) (NAUBEG(I),I=1,NTRACO)
      GOTO 5
C
C TRACE NORMALISATION
C
 140  TYPE 2140
 2140 FORMAT(' TRACE NORMALIZATION')
      IF(IPFLG.EQ.1) READ(1,1000) NRMFLG
      IF(IPFLG.EQ.0) TYPE 6140
 6140 FORMAT(' -1      To unit amplitude',/,
     +       '  0      To unit energy',/,
     +       '  1      To inverse energy',/,
     +       '$        Give option ? ')
      IF(IPFLG.EQ.0) ACCEPT 1000,NRMFLG
      WRITE(2,5000) NRMFLG
      GOTO 5
C
C  HILBERT TRANSFORM
C
 150  TYPE 2150
```

```
 2150 FORMAT(' REFLECTION STRENGTH')
      GOTO 5
C
C  STACK
C
 160  IF(IPFLG.EQ.0) GOTO 111
      READ(1,1000) ISTK,M,INTSW
      IF(ISTK.NE.2) READ(1,1400) NGATHR,(XSTART(I),I=1,NGATHR),XSTEP
 1400 FORMAT(I6,7F10.0)
      IF(ISTK.EQ.-2)        TYPE 2161
      IF(IABS(ISTK).EQ.1)   TYPE 2162
      IF(ISTK.LE.1)         TYPE 2163
      IF(ISTK.GE.1)         TYPE 2164
 2161 FORMAT('$REMOVE')
 2162 FORMAT('$APPLY')
 2163 FORMAT('  NMO CORRECTION')
 2164 FORMAT(' STACK')
      IF(NGATHR.EQ.1.AND.XSTART(1).LT.0.0) TYPE 2165
 2165 FORMAT(' SOURCE/RECEIVER OFFSETS TAKEN FROM TRACE HEADER')
      GOTO 112
 111  TYPE 6160
 6160 FORMAT(' NMO AND STACK',/,
     +' -2     Remove NMO correction',/,
     +' -1     Apply NMO correction',/,
     +'  1     Apply NMO correction and stack',/,
     +'  2     Stack only',/,
     +'$       Give option number ? ')
      ACCEPT 1000,ISTK
      M=1
      INTSW=0
      IF(ISTK.EQ.2) GOTO 112
      TYPE 6161
 6161 FORMAT('$Give interpolation level (1,2 or 4) ? ')
      ACCEPT 1000,M
      TYPE 6162
 6162 FORMAT('$Interpolate between velocity functions (Y/N) ? ')
      ACCEPT 3000,IANS
      IF(IANS.EQ.'Y') INTSW=1
      IF(NOFF.EQ.1) GOTO 112
      NOFF=1
      TYPE 6163
 6163 FORMAT('$Are offsets to be taken from trace header (Y/N) ? ')
      ACCEPT 3000,IANS
      IF(IANS.EQ.'Y') GOTO 112
      TYPE 6164
 6164 FORMAT('$Give number of different source/receiver no.1 offsets',
     +' (integer) ? ')
      ACCEPT 1000,NGATHR
      TYPE 6165,NGATHR
 6165 FORMAT(' Give',I2,' source/receiver no.1 offset(s) m (real)')
      ACCEPT 1100,(XSTART(I),I=1,NGATHR)
      TYPE 6166
 6166 FORMAT('$Give receiver spacing within gather m (real) ? ')
      ACCEPT 1100,XSTEP
 112  L=1
 161  IF(L.GE.M) GOTO 163
      L=L*2
      GOTO 161
 163  M=L
 162  RSUM=FLOAT(M*NSAMP2+24)
      IF(RSUM.LE.FLOAT(1E-IB)) GOTO 164
      IF(M.EQ.1) STOP'O/P TRACES TOO LONG'
      M=M/2
      TYPE 2166,M
 2166 FORMAT(/,' * WARNING * STACK INTERPOLATION LEVEL REDUCED TO ',
     +I3,/)
      GOTO 162
 164  WRITE(2,5000) ISTK,M,INTSW
      IF(ISTK.GT.0) NTRACO=1
      IF(ISTK.EQ.2) GOTO 5
      WRITE(2,5400) NGATHR,(XSTART(I),I=1,NGATHR),XSTEP
 5400 FORMAT(I5,',',7(F9.4,','))
```

```
      IST=1
      GOTO 5
C
C  THREE TRACE MIX
C
 170  TYPE 2170
 2170 FORMAT(' THREE TRACE MIX')
      NFILES=NFILES+1
      IF(NFILES.LT.4) STOP'TOO FEW CDPS TO TRACE MIX'
      IF(IPFLG.EQ.0) GOTO 171
      READ(1,1100) R(1),R(2),R(3)
      GOTO 172
 171  TYPE 6170
 6170 FORMAT('$Give mixing ratio (real) ?,?,? ')
      ACCEPT 1100,R(1),R(2),R(3)
 172  WRITE(2,5100) R(1),R(2),R(3)
      GOTO 5
C
C  VELOCITY FILTER
C
 180  TYPE 2180
 2180 FORMAT('$VELOCITY')
      TYPE 2091
      IF(IPFLG.EQ.0) GOTO 185
      READ(1,1000) NOP,IVELSH,M,INTSW
      READ(1,1400) NGATHR,(XSTART(I),I=1,NGATHR),XSTEP
      IF(NGATHR.EQ.1.AND.XSTART(1).LT.0.0) TYPE 2165
      GOTO 187
 185  TYPE 6180
 6180 FORMAT('$Give operator dimension <29 (integer) ? ')
      ACCEPT 1000,NOP
      TYPE 6181
 6181 FORMAT(' Give precentage shift on cut-off velocity functions',/,
     +' <100% filters slower coherent apparent arrivals',/,
     +' >100% filters faster coherent apparent arrivals',/,
     +'$(integer) ? ')
      ACCEPT 1000,IVELSH
      TYPE 6161
      ACCEPT 1000,M
      INTSW=0
      TYPE 6162
      ACCEPT 3000,IANS
      IF(IANS.EQ.'Y') INTSW=1
      IF(NOFF.EQ.1) GOTO 187
      NOFF=1
      TYPE 6163
      ACCEPT 3000,IANS
      IF(IANS.EQ.'Y') GOTO 187
      TYPE 6164
      ACCEPT 1000,NGATHR
      TYPE 6165,NGATHR
      ACCEPT 1100,(XSTART(I),I=1,NGATHR)
      TYPE 6166
      ACCEPT 1100,XSTEP
 187  NOP=(2*IFIX(NOP/2))+1
      L=1
 181  IF(L.GE.M) GOTO 182
      L=L*2
      GOTO 181
 182  M=L
 183  RDAT=FLOAT(NSAMP)*FLOAT(NOP)
      IF(RDAT.GT.24576.0) GOTO 186
 184  RSUM=RDAT+FLOAT((M*NSAMP2)+2)+FLOAT(NSAMP+NOP+(NOP*NOP)+22)
      IF(RSUM.LE.24576.0) GOTO 188
      IF(M.EQ.1) GOTO 186
      M=M/2
      TYPE 2181,M
 2181 FORMAT(/,' * WARNING * VEL. FILTER INTERP. LEVEL REDUCED TO ',
     +I3,/)
      GOTO 184
 186  NOP=NOP-2
      TYPE 2182,NOP
```

```
 2182 FORMAT(/,' * WARNING * VEL. FILTER OPERATOR SIZE REDUCED TO ',
     +I3,/)
      IF(NOP.LE.3) STOP'O/P TRACES TOO LONG'
      GOTO 183
 188  WRITE(2,5000) NOP,IVELSH,M,INTSW
      WRITE(2,5400) NGATHR,(XSTART(I),I=1,NGATHR),XSTEP
      IVF=1
      GOTO 5
C
C  DIP FILTER
C
 190  TYPE 2190
 2190 FORMAT('$DIP')
      TYPE 2091
      IF(IPFLG.EQ.0) GOTO 193
      READ(1,1400) IOP,XCMP,VAP
      GOTO 194
 193  TYPE 6180
      ACCEPT 1000,IOP
      TYPE 6190
 6190 FORMAT('$Give CMP spacing m (real) ? ')
      ACCEPT 1100,XCMP
      TYPE 6191
 6191 FORMAT('$Give minimum apparent velocity allowed km/sec',
     +' (real) ? ')
      ACCEPT 1100,VAP
 194  IOP=(2*IFIX(IOP/2))+1
 191  RDAT=FLOAT(NSAMP)*FLOAT(IOP)+FLOAT(NSAMP+IOP+(IOP*IOP))
      IF(RDAT.LE.24576.0) GOTO 192
      IOP=IOP-2
      TYPE 2191,IOP
 2191 FORMAT(/,' * WARNING * DIP FILTER OPERATOR SIZE REDUCED TO ',
     +I3,/)
      GOTO 191
 192  WRITE(2,5400) IOP,XCMP,VAP
      GOTO 5
C
C  TIME VARYING BAND-PASS FILTER
C
 200  TYPE 2200
 2200 FORMAT('$TIME VARYING BAND-PASS')
      TYPE 2091
      IF(IPFLG.EQ.0) GOTO 201
      READ(1,1400) ITM1,FL1,FH1
      IF(ITM1.GE.0) GOTO 202
      TYPE 2201,IABS(ITM1)
 2201 FORMAT(' BAND-PASS CONSTANT UNTIL ',I5,' MSEC BELOW SEA-FLOOR')
 202  READ(1,1400) ITM2,FL2,FH2
      IF(ITM2.GE.0) GOTO 204
      TYPE 2202,IABS(ITM2)
 2202 FORMAT(' BAND-PASS CONSTANT AFTER ',I5,' MSEC BELOW SEA-FLOOR')
      GOTO 204
 201  TYPE 6200
 6200 FORMAT(' The band-pass remains constant until the first time',
     +' given',/,' Then linearly varies until the second time given ',
     +/,' Give first time',/,'$ -ve if time represents delay',
     +' after sea-floor msec (integer) ? ')
      ACCEPT 1400,ITM1
      IF(ITM1.LT.0) IDFLG=IDFLG+1
      TYPE 6090
      ACCEPT 1100,FL1,FH1
      TYPE 6201
 6201 FORMAT(' Give second time',/,'$ -ve if time represents delay',
     +' after sea-floor msec (integer) ? ')
      ACCEPT 1400,ITM2
      IF(ITM2.LT.0) IDFLG=IDFLG+1
      TYPE 6090
      ACCEPT 1100,FL2,FH2
 204  IF(FL1.GT.FSAMP.OR.FH1.GT.FSAMP
     +    .OR.FL2.GT.FSAMP.OR.FH2.GT.FSAMP) GOTO 203
      IF(ITM1.GT.-10000) WRITE(2,5400) ITM1,FL1,FH1
      IF(ITM2.GT.-10000) WRITE(2,5400) ITM2,FL2,FH2
```

```
      IF(ITM1.LE.-10000) WRITE(2,5500) ITM1,FL1,FH1
      IF(ITM2.LE.-10000) WRITE(2,5500) ITM2,FL2,FH2
 5500 FORMAT(I6,2(F9.4,','))
      GOTO 5
 203  IF(IPFLG.NE.0) STOP'I/P FREQUENCY IN HZ'
      TYPE 6093
      GOTO 200
C
C  HORIZONTAL SUM OF STACKED TRACES
C
 210  TYPE 2210
 2210 FORMAT(' HORIZONTAL TRACE SUM FOR STACKED DATA')
      IF(IPFLG.EQ.0) GOTO 211
      READ(1,1000) NHSUM
      GOTO 212
 211  TYPE 6210
 6210 FORMAT('$Give number of stacked traces to sum (integer) ? ')
      ACCEPT 1000,NHSUM
 212  WRITE(2,5000) NHSUM
      GOTO 5
C
C  AUTO STATICS
C
 220  TYPE 2220
 2220 FORMAT(' AUTO STATICS ON NMO''ED UNSTACKED GATHER')
      IF(IPFLG.EQ.0) GOTO 221
      READ(1,1000) IPSTR,IPEND
      READ(1,1000) IXSTR,IXLEN,MAXSH,IDOC
      IF(NTRACO.GT.24) IDOC=0
      GOTO 222
 221  TYPE 6220
 6220 FORMAT(' Give first and last processed trace numbers',/,
     +'$to include in pilot trace (integer) ?,? ')
      ACCEPT 1000,IPSTR,IPEND
      TYPE 6221
 6221 FORMAT(' Give start time and window length for cross correlation',
     +/,'$function msec (integer) ?,? ')
      ACCEPT 1000,IXSTR,IXLEN
      TYPE 6222
 6222 FORMAT('$Give maximum shift allowed on each trace msec',
     +' (integer) ? ')
      ACCEPT 1000,MAXSH
      IDOC=0
      IF(NTRACO.GT.24) GOTO 222
      TYPE 6223
 6223 FORMAT('$Include sample shift applied to each trace',
     +' in log file (Y/N) ? ')
      ACCEPT 3000,IANS
      IF(IANS.EQ.'Y') IDOC=1
 222  WRITE(2,5000) IPSTR,IPEND
      WRITE(2,5000) IXSTR,IXLEN,MAXSH,IDOC
      GOTO 5
C
C      EXIT CALL
C
 250  TYPE 2250
 2250 FORMAT(' EXIT AND OUTPUT TRACE TO TAPE')
C
C  GET ASCII INFORMATION TO GO IN REEL HEADER
C
      IF(IPFLG.EQ.0) GOTO 255
      READ(1,1000) ILINE
      IF(ILINE.LE.0) GOTO 256
      IF(ILINE.GT.10) STOP'ILLEGAL NO. OF TEXT LINES'
      BACKSPACE 1
      READ(1,3200) ILINE,((ITEXT(I,J),I=1,64),J=1,ILINE)
 3200 FORMAT(I6,(64A1))
      GOTO 256
 255  TYPE 6250
 6250 FORMAT(/,'$Give number of lines of text (integer) ? ')
      ACCEPT 1000,ILINE
      IF(ILINE.LE.0) GOTO 256
```

```
      TYPE 6251
 6251 FORMAT(' Give text, max. 64 characters per line',/,
     +' |<START',51X,'STOP>|')
      ACCEPT 3201,((ITEXT(I,J),I=1,64),J=1,ILINE)
 3201 FORMAT(64A1)
 256  IF(ILINE.LE.0) WRITE(2,5000) ILINE
      IF(ILINE.GE.1) WRITE(2,5200) ILINE,((ITEXT(I,J),I=1,64),J=1,ILINE)
 5200 FORMAT(I5,',',(64A1))
      IF(IPFLG.EQ.1) CLOSE(UNIT=1)
      CLOSE(UNIT=2)
C
C  FLAG DATA FILES EXPECTED
C
      IF(IST.EQ.1) TYPE 2167
 2167 FORMAT(/,' | NOTICE | EXPECTS STACKING VELOCITY DATA IN',
     +' DK6:VELRMS.DAT',/)
      IF(IVF.EQ.1) TYPE 2184
 2184 FORMAT(/,' | NOTICE | EXPECTS VELOCITY FILTER DATA IN',
     +' DK6:VELFIL.DAT',/)
      IF(IDFLG.GT.0) TYPE 3700
 3700 FORMAT(/,' | NOTICE | DK6:DEPTH.DAT FILE REQURIED',/,
     +' UNLESS PROCESSING STACKED DATA WHEN FIRST TWO-WAY TRAVEL',/,
     +' TIME OF VELOCITY FUNCTION WILL BE USED FROM TRACE HEADER',/)
C
C  EXIT PROGRAM IF DATA FILE WRONG
C
      IF(IPFLG.EQ.0) GOTO 450
      TYPE 4400
 4400 FORMAT(/,'$PROCESSING ORDER OK ? (Y/N) ')
      ACCEPT 3000,IANS
 3000 FORMAT(A1)
      IF(IANS.NE.'Y') STOP'INPUT DATA FILE ERROR'
C
C  DATA FILE O.K.
C
 450  TYPE 4500
 4500 FORMAT(/,'$CHAIN SEISMIC PROCESSING PACKAGE (Y/N) ? ')
      ACCEPT 3000,IANS
      IF(IANS.NE.'Y') STOP'DATA FILE WRITTEN TO DK6:SEIS.DAT'
      CALL RESET
      CALL CHAIN(SEIS,,0)
      STOP
      END
```

```
C
C  THIS IS AN INTERACTIVE VELOCITY INPUT ROUTINE THAT OUTPUTS
C  A DATA FILE FOR THE PROCESSING PROGRAM.
C  THE PROGRAM CREATES OR EDITS THE DATA FILE ALLOWING
C  THE INSERTING OF A CMP
C  THE DELETING OF A CMP
C  THE CHANGING OF A CMP
C  THE EXAMINING OF A CMP
C
C  WRITTEN BY: R.W.HOBBS FEBRUARY 1984
C
      VIRTUAL V(2,11,100)
      REAL*8 NAME(2),NNEW
      INTEGER*2 SPEC(39),EXT(4),DCIN(4),DCOUT(4)
      LOGICAL*1 REP,FNAME(16),IERR
C
      EQUIVALENCE (SPEC(16),NNEW),(DCIN(1),NAME(1)),(DCOUT(1),NAME(2))
C
      DATA NAME(2)/12RDK6VELDATOLD/,EXT/4*0/
C
C  CHECK FOR RE-EDITTING EXISTING STACK DATA FILE
C
      NVEL=0
      IFLG=0
      TYPE 2000
 2000 FORMAT(' INTERACTIVE INPUT PROGRAM FOR VELOCITY/TIME DATA'.//.
     +'$Re-editting existing data file (Y/N) ? ')
      ACCEPT 1500,REP
 1500 FORMAT(A1)
      IF(REP.EQ.'Y') TYPE 2100
      IF(REP.EQ.'N') TYPE 2200
 2100 FORMAT(/,'$Give full name of input/output file ? ')
 2200 FORMAT(/,'$Give full name of output file ? ')
      CALL GETSTR(5,FNAME,14,IERR)
      IF(IERR.NE..FALSE..OR.FNAME(1).EQ.0) STOP'ILLEGAL FILE NAME'
      IF(REP.EQ.'N') GOTO 400
C
C  OPEN EXISTING FILE
C
      OPEN(UNIT=1,NAME=FNAME,TYPE='OLD',ERR=10)
      GOTO 20
 10   STOP'NO INPUT FILE FOUND'
C
C  READ ANALYSES
C
 20   CONTINUE
 50   READ(1,1000,END=70) NCMP,NLYR
 1000 FORMAT(12I6)
      NVEL=NVEL+1
      V(1,1,NVEL)=FLOAT(NCMP)
      V(2,1,NVEL)=FLOAT(NLYR)
      DO 60 K=1,NLYR
 60   READ(1,1100) V(1,K+1,NVEL),V(2,K+1,NVEL)
 1100 FORMAT(3F10.0)
      GOTO 50
C
C  CLOSE DATA FILE AND RENAME AS *:VELDAT.OLD
C
 70   CLOSE(UNIT=1)
      IF(ICSI(SPEC,EXT,FNAME,,0).NE.0) STOP'STRING ERROR'
      NAME(1)=NNEW
      DCOUT(1)=DCIN(1)
      ICHN=IGETC()
      IF(ICHN.LT.0) STOP'NO CHANNEL'
      CALL IRENAM(ICHN,NAME)
      TYPE 2300
 2300 FORMAT(' INPUT FILE HAS BEEN RENAMED TO *:VELDAT.OLD')
      CALL IFREEC(ICHN)
      GOTO 400
C
C START EDITTING VELOCITY ANALYSES
C
```

```
 400  TYPE 4000
 4000 FORMAT(///,' VELOCITY ANALYSIS CMP OPTIONS',//,
     +' (I)nsert CMP',/,
     +' (D)elete CMP',/,
     +' (C)hange CMP',/,
     +' (E)xamine CMP',/,
     +' (P)rint velocity file',/,
     +' (V)erify layer number for lateral interpolation between CMPS',/,
     +' (L)ist CMP numbers',/,
     +' e(X)it and write out data file',//,
     +'$     Give option letter ? ')
      ACCEPT 1500,REP
C
C  CHECK FOR VALID COMMAND
C
      IF(    REP.EQ.'I'
     +    .OR.REP.EQ.'D'
     +    .OR.REP.EQ.'C'
     +    .OR.REP.EQ.'E'
     +    .OR.REP.EQ.'P'
     +    .OR.REP.EQ.'V'
     +    .OR.REP.EQ.'L'
     +    .OR.REP.EQ.'X') GOTO 410
      TYPE 4050
 4050 FORMAT(/,' ILLEGAL OPTION. USE KEY LETTER ONLY INDICATED BY ( )')
      GOTO 400
 410  IF(REP.EQ.'X') GOTO 990
      IF(REP.EQ.'L') GOTO 900
      IF(REP.EQ.'V') GOTO 850
      IF(REP.EQ.'P') GOTO 800
C
C  GET CMP NUMBER
C
      IF(NVEL.EQ.0.AND.REP.NE.'I') GOTO 400
      TYPE 4100
 4100 FORMAT(/,'$Give CMP number ? ')
      ACCEPT 1000,LCMP
C
C  INDENTIFY EDIT COMMANDS
C
      IF(REP.EQ.'I') GOTO 500
      DO 430 I=1,NVEL
 430  IF(LCMP.EQ.IFIX(V(1,1,I))) GOTO 440
      TYPE 4200,LCMP
 4200 FORMAT(' CMP NO:',I6,' NOT FOUND')
      GOTO 400
 440  IF(REP.EQ.'D') GOTO 600
      IF(REP.EQ.'C') GOTO 700
      IF(REP.EQ.'E') GOTO 700
      GOTO 400
C
C  INSERT CMP
C
 500  I=1
      IF(NVEL.EQ.0) GOTO 540
      DO 510 I=1,NVEL
 510  IF(LCMP.LE.IFIX(V(1,1,I))) GOTO 520
      I=NVEL+1
      GOTO 540
 520  IF(LCMP.EQ.IFIX(V(1,1,I))) GOTO 700
      DO 530 J=NVEL,I,-1
      DO 530 K=1,11
      DO 530 L=1,2
 530  V(L,K,J+1)=V(L,K,J)
 540  NVEL=NVEL+1
      TYPE 5000
 5000 FORMAT('$Give number of layers ? ')
      ACCEPT 1000,NLYR
      IF(NLYR.LE.0) GOTO 600
      IF(NLYR.LE.10) GOTO 542
      TYPE 5050
 5050 FORMAT(' TOO MANY LAYERS, MAXIMUM OF 10 ALLOWED')
```

```
      GOTO 540
 542  V(1,1,I)=FLOAT(LCMP)
      V(2,1,I)=FLOAT(NLYR)
      TYPE 5100
 5100 FORMAT(/,' Give time/velocity data as either',/,
     +' Two-way travel time msec, RMS velocity km/sec <CR>',/,
     +' Two-way travel time msec,, interval velocity km/sec <CR>')
      DO 560 K=1,NLYR
      ACCEPT 1100,V(1,K+1,I),V(2,K+1,I),VINT
      IF(K.EQ.1) GOTO 545
      IF(V(1,K+1,I).GT.V(1,K,I)) GOTO 545
      TYPE 5400
 5400 FORMAT(/,' LAYERS MUST BE IN TIME INCREASING ORDER')
      GOTO 600
 545  IF(V(2,K+1,I).GT.0.0) GOTO 560
      IF(K.GE.2) GOTO 550
      V(2,K+1,I)=VINT
      GOTO 560
 550  V2=((V(2,K,I)*V(2,K,I)*V(1,K,I))+
     +(VINT*VINT*(V(1,K+1,I)-V(1,K,I))))/V(1,K+1,I)
      IF(V2.GT.0.0) GOTO 555
      TYPE 5500,K
 5500 FORMAT(' LAYER',I3,' HAS NEGATIVE RMS VELOCITY, SET TO 0.0')
      V2=0.0
 555  V(2,K+1,I)=SQRT(V2)
 560  CONTINUE
      GOTO 700
C
C  DELETE CMP
C
 600  DO 610 J=I,NVEL-1
      DO 610 K=1,11
      DO 610 L=1,2
 610  V(L,K,J)=V(L,K,J+1)
      NVEL=NVEL-1
      GOTO 400
C
C  CHANGE/EXAMINE CMP
C
 700  NLYR=IFIX(V(2,1,I))
      TYPE 7100,LCMP,NLYR
 7100 FORMAT(///,' CMP NO:',I5,' HAS ',I2,' LAYERS',//,
     +' LAYER TWO-WAY TIME  RMS VELOCITY  INT VELOCITY')
      DO 730 K=1,NLYR
      IF(K.GE.2) GOTO 731
      VINT=V(2,K+1,I)
      GOTO 734
 731  IF(V(1,K+1,I).GT.V(1,K,I)) GOTO 732
      TYPE 7125
 7125 FORMAT(/,' * WARNING * TWO-WAY TRAVEL TIME NOT IN ACCENDING ORDER
     + INT. VEL. SET TO 0.0',/)
      VINT=0.0
      GOTO 734
 732  V2=((V(2,K+1,I)*V(2,K+1,I)*V(1,K+1,I))-
     +(V(2,K,I)*V(2,K,I)*V(1,K,I)))/(V(1,K+1,I)-V(1,K,I))
      IF(V2.GT.0.0) GOTO 733
      TYPE 7150,K
 7150 FORMAT(' LAYER',I3,' HAS NEGATIVE INT VELOCITY, SET TO 0.0')
      V2=0.0
 733  VINT=SQRT(V2)
 734  TYPE 7200,K,V(1,K+1,I),V(2,K+1,I),VINT
 730  CONTINUE
 7200 FORMAT(' ',1X,I2,5X,F7.0,7X,F6.2,8X,F6.2)
C
C  IF ONLY EXAMINE RETURN TO START
C
      IF(REP.EQ.'E') GOTO 400
C
C  EDIT LAYERS
C
 745  TYPE 7300
 7300 FORMAT(///,' LAYER OPTIONS',//,
```

```
     +' (I)nsert layer',/,
     +' (D)elete layer',/,
     +' (C)hange layer',/,
     +' (A)ccept CMP',//,
     +'$     Give option letter ? ')
      ACCEPT 1500,REP
C
C  CHECK FOR VALID COMMAND
C
      IF(    REP.EQ.'I'
     +   .OR.REP.EQ.'D'
     +   .OR.REP.EQ.'C'
     +   .OR.REP.EQ.'A') GOTO 740
      TYPE 4050
      GOTO 745
 740  IF(REP.EQ.'A') GOTO 400
      TYPE 7400
 7400 FORMAT(/,'$Give layer number ? ')
      ACCEPT 1000,LYR
      IF(LYR.LE.0) GOTO 700
      IF(REP.EQ.'I') GOTO 750
      IF(LYR.GT.NLYR) GOTO 700
      IF(REP.EQ.'D') GOTO 760
      IF(REP.EQ.'C') GOTO 752
      TYPE 4050
      GOTO 700
C
C  INSERT/CHANGE LAYER
C
 750  IF(LYR.GT.(NLYR+1)) GOTO 700
      IF(NLYR.LE.9) GOTO 758
      TYPE 5050
      GOTO 700
 758  IF(LYR.GT.NLYR) GOTO 752
      DO 751 K=NLYR,LYR,-1
      DO 751 L=1,2
 751  V(L,K+2,I)=V(L,K+1,I)
 752  TYPE 5100
      ACCEPT 1100,V(1,LYR+1,I),V(2,LYR+1,I),VINT
      IF(LYR.GE.NLYR) GOTO 757
      IF(V(1,LYR+1,I).GT.V(1,LYR+2,I)) GOTO 756
 757  IF(LYR.EQ.1) GOTO 755
      IF(V(1,LYR+1,I).GT.V(1,LYR,I)) GOTO 755
 756  TYPE 5400
      IF(REP.EQ.'I') NLYR=NLYR+1
      GOTO 760
 755  IF(V(2,LYR+1,I).GT.0.0) GOTO 754
      IF(LYR.GE.2) GOTO 753
      V(2,LYR+1,I)=VINT
      GOTO 754
 753  V2=((V(2,LYR,I)*V(2,LYR,I)*V(1,LYR,I))+
     +(VINT*VINT*(V(1,LYR+1,I)-V(1,LYR,I))))/V(1,LYR+1,I)
      IF(V2.GT.0.0) GOTO 759
      TYPE 5500,LYR
      V2=0.0
 759  V(2,LYR+1,I)=SQRT(V2)
 754  IF(REP.EQ.'C') GOTO 700
      NLYR=NLYR+1
      V(2,1,I)=FLOAT(NLYR)
      GOTO 700
C
C  DELETE A LAYER
C
 760  DO 761 K=LYR,NLYR-1
      DO 761 L=1,2
 761  V(L,K+1,I)=V(L,K+2,I)
      NLYR=NLYR-1
      IF(NLYR.LE.0) GOTO 600
      V(2,1,I)=FLOAT(NLYR)
      GOTO 700
C
C  PRINT OUT VELOCITY DATA
```

```
C
 800  IF(NVEL.EQ.0) GOTO 400
      TYPE 8100
 8100 FORMAT(/,' TYPE <CTRL>C TO STOP PRINT AND RETURN TO BASIC MENU',/)
      IFLAG=0
      CALL SCCA(IFLAG)
      DO 810 J=1,NVEL
      LCMP=IFIX(V(1,1,J))
      NLYR=IFIX(V(2,1,J))
      PRINT 7100,LCMP,NLYR
      DO 810 K=1,NLYR
      IF(K.GE.2) GOTO 811
      VINT=V(2,K+1,J)
      GOTO 814
 811  IF(V(1,K+1,J).GT.V(1,K,J)) GOTO 812
      TYPE 7125
      VINT=0.0
      GOTO 814
 812  V2=((V(2,K+1,J)*V(2,K+1,J)*V(1,K+1,J))-
     +(V(2,K,J)*V(2,K,J)*V(1,K,J)))/(V(1,K+1,J)-V(1,K,J))
      IF(V2.GT.0.0) GOTO 813
      PRINT 7150,K
      V2=0.0
 813  VINT=SQRT(V2)
 814  PRINT 7200,K,V(1,K+1,J),V(2,K+1,J),VINT
      IF(ITTINR().NE.3) GOTO 810
      IF(IFLAG.EQ.0) GOTO 820
      GOTO 990
 810  CONTINUE
      PRINT 8000
      PRINT 8000
 8000 FORMAT('1')
 820  CLOSE(UNIT=6)
      CALL SCCA
      GOTO 400
C
C  CHECK FOR LAYER CHANGES AT NON-AJACENT CMPS
C
 850  IF(NVEL.LE.1) GOTO 400
      TYPE 8500
 8500 FORMAT(/,' CHANGE IN LAYER NUMBER IS ONLY ALLOWED WHEN',/,
     +' TWO VELOCITY ANALYSES ARE AT AJACENT CMPS',/)
      IVFLG=0
      DO 860 J=1,NVEL-1
      IF(V(2,1,J).EQ.V(2,1,J+1)) GOTO 860
      IF((V(1,1,J+1)-V(1,1,J)).LE.1.0) GOTO 860
      LCMP=IFIX(V(1,1,J))
      NCMP=IFIX(V(1,1,J+1))
      IVFLG=IVFLG+1
      TYPE 8600,LCMP,NCMP
 8600 FORMAT(' LAYER NUMBER CHANGE BETWEEN CMP :',I5,' AND CMP :',I5)
 860  CONTINUE
      IF(IVFLG.EQ.0) TYPE 8601
      IF(IVFLG.GT.0) TYPE 8602,IVFLG
 8601 FORMAT(' NO ILLEGAL LAYER NUMBER CHANGES')
 8602 FORMAT(' THERE ARE',I4,' ILLEGAL LAYER NUMBER CHANGES')
      GOTO 400
C
C  LIST ALL CMPS
C
 900  IF(NVEL.EQ.0) GOTO 400
      TYPE 3900
 3900 FORMAT(/)
      TYPE 3910,(IFIX(V(1,1,J)),J=1,NVEL)
 3910 FORMAT((' ',12(I5,',')))
      GOTO 400
C
C  EXIT AND UPDATE
C
 990  OPEN(UNIT=1,NAME=FNAME,TYPE='NEW',ERR=991)
      GOTO 992
 991  STOP'OUTPUT DATA FILE OPEN ERROR'
```

```
992   IF(NVEL.EQ.0) GOTO 999
      DO 995 I=1,NVEL
      NCMP=IFIX(V(1,1,I))
      NLYR=IFIX(V(2,1,I))
      WRITE(1,9000) NCMP,NLYR
9000  FORMAT(12(I5,','))
      DO 993 K=1,NLYR
993   WRITE(1,9100) V(1,K+1,I),V(2,K+1,I)
9100  FORMAT(2(F9.3,','))
995   CONTINUE
999   CLOSE(UNIT=1)
      STOP'OUTPUT DATA FILE WRITTEN'
      END
```

```
C  THIS IS THE ROOT CONTROL SEGMENT OF THE SEISMIC PROCESSING
C  PACKAGE. INPUT PARAMETERS ARE READ FROM A DATA FILE DK6:SEIS.DAT
C  (USE PRCDAT TO CREATE OR VERIFY) BY THE INPUT SUBROUTINE
C  SEISIN. DEFAULT VALUES, VELOCITY/TIME DATA  AND DEPTH DATA FILES
C  ARE SET-UP AND/OR READ BY SUBROUTINE SEISIO, THIS ALSO READS THE
C  TAPE HEADER BLOCKS FROM INPUT TAPE. SEISHD SUBROUTINE UP-DATES THE
C  TAPE HEADER BLOCKS, STORED ON A DISK FILE DK6:SEISHD.DAT, AND
C  WRITES THEM TO THE OUTPUT TAPE. THE PROCESSING WORK IS DONE
C  BY THE SEISSB SUBROUTINE, THIS CALLS ALL THE PROCESSING OPTION
C  ROUTINES. SEISND TIDIES UP ALL OPEN OUTPUT FILES AND WRITES OUT
C  END OF RUN PARAMETERS INCLUDING ANY ERROR MESSAGES.
C
C  INPUT:            DATA FILE DK6:SEIS.DAT
C                    DURHAM SEG-Y DATA TAPE
C  OUTPUT:           LOG FILE DK6:SEIS.LOG
C                    DURHAM SEG-Y DATA TAPE
C                    UP TO TWO AUTOCORRELATION DATA DISK FILES
C                          DR6:AUTO1.DAT
C                          DR6:AUTO2.DAT
C
C  WRITTEN R.W.HOBBS 1984
C  BASED ON ORIGINAL PROGRAM BY M.J.POULTER THESIS 1982
C        AND SUBSEQUENT WORK BY R.W.HOBBS, D.W.WELLBY AND M.J.SMITH
C
      VIRTUAL ICDPED(100),ITRCED(100),ITRACP(48),IDEPTH(10000),
     +ITEXT(64,10),NBEG(3,96),NAUBEG(2,96),V(2,11,100),
     +X(96),VF(2,11,100),SEIS(4156),XSTART(6),RLHDBL(80,40),
     +RLDUM(200)
      REAL*4 TMNAM(3)
      INTEGER*2 IORD(12,20),ITMERR(2),REELNU,TPDRR,TPDRW,RLDUM
      LOGICAL*1 OUTBUF(240),ITEXT,RLHDBL
C
      COMMON/INPUT/ NFILES,NCHAN,NSMPIN,ISTART,ILEN,IFIRST,ILAST,
     +FTRCNO,IWORDR,IWORDW,IBLK,IVFRT,IDPRT,NTRACI
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/CTRL/  ITM,IST,IVF,IDP,IAU,IORD
      COMMON/STATDP/IDFLG,IDPTOT,IDPINT,ISSHOT,IXES,IXSR,IXSS,IXRR
      COMMON/DATBUF/OUTBUF
      COMMON/REELS/ ILINE,REELNU,NTRACO
      COMMON/TAPE/  TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      DATA DRHAN/3RDR /,TMHAN/3RTM /,APHAN/3RAP /
      DATA TMNAM(1)/3RTM4/,TMNAM(2)/3RTM5/,TMNAM(3)/3RTM6/
C
C  SET UP I/O
C
      IF(IFETCH(DRHAN).NE.0) STOP 'DR FETCH ERR'
      IF(IFETCH(APHAN).NE.0) STOP 'AP FETCH ERR'
      IF(IFETCH(TMHAN).NE.0) STOP 'TAPE FETCH ERR'
      CALL APINIT(0,0,1)
C
C  CALL INPUT SUBROUTINE
C
      CALL SEISIN(ICDPED,ITRCED,ITRACP,NBEG,NAUBEG,X,XSTART,ITEXT)
      CALL SEISIO(SEIS,IDEPTH,V,VF,RLHDBL,RLDUM)
      ISTOP=0
C
C  LOOKUP WRITE TAPE DRIVE
C
      CALL SEISHD(ITEXT,ICDPNO,1,ISTOP)
      IF(ISTOP.NE.0) CALL SEISND(ISTOP,ICDPNO)
C
C  DO PROCESSING
C
      CALL SEISSB(SEIS,ICDPED,ITRCED,ITRACP,IDEPTH,
     +NBEG,NAUBEG,V,VF,X,ITEXT,ICDPNO,ISTOP)
      CALL SEISND(ISTOP,ICDPNO)
      CALL APRLSE
      CALL RESET
      STOP
      END
```

```
      FUNCTION ISMPLE(NTIME,FSAMP,NDELAY)
C
C  TO CONVERT ABSOLUTE TIME IN MILLI-SEC TO RELATIVE TRACE SAMPLE NOS.
C
      ISMPLE=IFIX(FLOAT(NTIME-NDELAY)*FSAMP)
      RETURN
      END


      BLOCK DATA
C
      INTEGER*2 IORD(12,20)
C
      COMMON/CTRL/  ITM,IST,IVF,IDP,IAU,IORD
C
      DATA ITM,IST,IVF,IDP,IAU/0,0,0,0,0/
      DATA IORD(1,1)/240*0/
C
      END




C  INPUT SUBROUTINE FOR THE SEISMIC PROCESSING PACKAGE.
C  READS INPUT PARAMETERS FROM THE DATA FILE DK6:SEIS.DAT AND LOADS
C  UP AN ARRAY IORD(X,Y) WHICH IS USED TO CONTROL ALL PROCESSING
C  OPTIONS.
C
C  IORD(1,Y)  THIS IS THE OPTION 'NAME' NUMBER, WHERE Y REPRESENTS
C             THE SEQUENCE NUMBER WHEN THE GIVEN PROCESS IS DONE TO
C             THE INPUT TRACE
C
      SUBROUTINE SEISIN(ICDPED,ITRCED,ITRACP,NBEG,NAUBEG,X,
     +XSTART,ITEXT)
C
      VIRTUAL ICDPED(100),ITRCED(100),ITRACP(48),ITEXT(64,10),
     +NBEG(3,96),NAUBEG(2,96),X(96),XSTART(6)
      REAL*8 FSPECA(2)
      REAL*4 TEMP(4),R(3),TMNAM(3)
      INTEGER*2 IORD(12,20),ITMERR(2),REELNU,TPDRR,TPDRW,
     +IEDLEV(2),IFACT(2),IR(6),IFNQ(2),ISNQ(2),IVAP(2),
     +JB1(2),IOM1(2),JB2(2),IOM2(2)
      LOGICAL*1 ITEXT
C
      COMMON/INPUT/ NFILES,NCHAN,NSMPIN,ISTART,ILEN,IFIRST,ILAST,
     +FTRCNO,IWORDR,IWORDW,IBLK,IVFRT,IDPRT,NTRACI
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/CTRL/  ITM,IST,IVF,IDP,IAU,IORD
      COMMON/STATDP/IDFLG,IDPTOT,IDPINT,ISSHOT,IXES,IXSR,IXSS,IXRR
      COMMON/REELS/ ILINE,REELNU,NTRACO
      COMMON/TAPE/  TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (TEMP(1),R(1)),(TEMP(1),B1),(TEMP(2),B2),
     +(TEMP(3),OM1),(TEMP(4),OM2),(TEMP(1),FNQ),(TEMP(2),SNQ),
     +(TEMP(3),VAP),(TEMP(1),EDLEV),(TEMP(2),FACT)
      EQUIVALENCE (IEDLEV(1),EDLEV),(IFACT(1),FACT),(IR(1),R(1)),
     +(IFNQ(1),FNQ),(ISNQ(1),SNQ),(IVAP(1),VAP),(JB1(1),B1),
     +(IOM1(1),OM1),(JB2(1),B2),(IOM2(1),OM2)
C
      DATA FSPECA(1)/12RDR6AUTO1 DAT/,FSPECA(2)/12RDR6AUTO2 DAT/
C
C
C  OPEN DATA FILE
C
      OPEN(UNIT=1,TYPE='OLD',NAME='DK6:SEIS.DAT',ERR=6)
      GOTO 7
 6    STOP 'NO DATA FILE'
 7    READ(1,1005)NFILES,NCHAN,NSMPIN,FSAMP
1000  FORMAT(12I6)
      READ(1,1000)IFIRST,ILAST,ISTART,ILEN
```

```
 1005 FORMAT(3I6,F10.0)
 1100 FORMAT(3F10.0)
      READ(1,1005)TPDRR,TPDRW,REELNU,FTRCNO
C
C  SET UP CONSTANTS
C
      NTRACO=ILAST-IFIRST+1
      NTRACI=NTRACO
      NSAMP=ISMPLE(ILEN,FSAMP,0)
      NSAMP2=2
 1    IF(NSAMP2.GE.NSAMP)GOTO 2
      NSAMP2=NSAMP2*2
      GOTO 1
 2    DFI=FSAMP/(2*NSAMP2)
      IP=0
      ISFLG=0
      ISIG=0
      IAUTO=0
      IDFLG=0
C
C READ IN JOB SPECIFIC DATA AND SET UP
C THE FILTERS TO BE USED
C
 5    IP=IP+1
      READ(1,1000) IORD(1,IP)
      GOTO (10,20,30,40,50,60,70,80,90,100,110,
     +120,130,140,150,160,170,180,190,200,210,
     +220,250,250,250) IORD(1,IP)
C
C  TIME BREAK CORRECTION
C
 10   READ(1,1000) IORD(2,IP)
      READ(1,1000) IORD(3,IP),IORD(4,IP)
      IF(IORD(4,IP).LE.0) GOTO 5
      DO 12 I=1,IORD(4,IP)
 12   READ(1,1000) IORD((I*2)+3,IP),IORD((I*2)+4,IP)
      GOTO 5
C
C TRACE EDIT DATA
C
 20   READ(1,1000) IORD(2,IP),IORD(3,IP)
      IORD(9,IP)=1
      IF(IORD(2,IP).EQ.0) GOTO 25
      DO 23 I=1,IORD(2,IP)
 23   READ(1,1000) ICDPED(I),ITRCED(I)
 25   IF(IORD(3,IP).EQ.0) GOTO 5
      READ(1,1005) IORD(4,IP),IORD(5,IP),IORD(6,IP),EDLEV
      IORD(5,IP)=ISMPLE(IORD(5,IP),FSAMP,0)
      DO 22 I=1,2
 22   IORD(6+I,IP)=IEDLEV(I)
      GOTO 5
C
C MUTE DATA
C
 30   READ(1,1000)IORD(2,IP),IORD(3,IP),IORD(4,IP)
      IORD(2,IP)=ISMPLE(IORD(2,IP),FSAMP,0)
      IF(IORD(4,IP).EQ.0.OR.IORD(3,IP).LT.0) GOTO 5
      DO 32 I=1,IORD(4,IP)
 32   READ(1,1000) IORD((I*2)+3,IP),IORD((I*2)+4,IP)
      GOTO 5
C
C POLARITY REVERSAL DATA
C
 40   READ(1,1000) IORD(2,IP)
      READ(1,1000)(ITRACP(I),I=1,IORD(2,IP))
      GOTO 5
C
C  STATICS
C
 50   IDFLG=IDFLG+1
      GOTO 5
C
```

```
C TIME RAMP DATA
C
 60   READ(1,1300) IORD(2,IP),IORD(3,IP),FACT
 1300 FORMAT(2I6,F10.0)
      DO 62 I=1,2
 62   IORD(3+I,IP)=IFACT(I)
      GOTO 5
C
C TV**2 RAMP
C
 70   READ(1,1000) IORD(2,IP)
      GOTO 5
C
C  AGC
C
 80   READ(1,1000) NWIN,IORD(3,IP)
      IORD(2,IP)=ISMPLE(NWIN,FSAMP*0.5,0)
      IF(IORD(3,IP).LT.0) GOTO 85
      IORD(3,IP)=ISMPLE(IORD(3,IP),FSAMP,0)
      IF(IORD(2,IP).GT.IORD(3,IP)) IORD(3,IP)=IORD(2,IP)
      GOTO 5
 85   IORD(2,IP)=-IORD(2,IP)
      IDFLG=IDFLG+1
      GOTO 5
C
C BANDPASS FILTER
C
 90   READ(1,1100)FL2,FH1
      READ(1,1100)FL1,FH2
      IORD(2,IP)=(2*IFIX((FL2-FL1)/DFI))+2
      IORD(7,IP)=IFIX(FL1/DFI)
      IORD(3,IP)=(2*IFIX(FH1/DFI))+2
      IORD(6,IP)=IFIX(FH2/DFI)
      IORD(4,IP)=IORD(3,IP)+(2*IORD(6,IP))
      IORD(5,IP)=(2*NSAMP2)+2-IORD(4,IP)
      IF(IORD(5,IP).LE.1) IORD(5,IP)=2
      GOTO 5
C
C BANDREJECT FILTER
C
 100  READ(1,1100)FL2,FH1
      FR=(FL2+FH1)/2.0
      IORD(3,IP)=2*IFIX(FR/DFI)+2
      IORD(4,IP)=IFIX((FR-FL2)/DFI)
      IORD(2,IP)=IORD(3,IP)-(2*IORD(4,IP))+2
      GOTO 5
C
C WIENER AND PREDICTION ERROR FILTER
C
 110  CONTINUE
 120  ISIG=ISIG+1
      IORD(2,IP)=ISIG
      READ(1,1000)IORD(3,IP),IORD(6,IP),IORD(4,IP),IORD(5,IP)
      IF(IORD(3,IP).EQ.3) GOTO 124
      READ(1,1000) NBEG(ISIG,1)
      GOTO 125
 124  READ(1,1000) (NBEG(ISIG,I),I=1,NTRACO)
 125  READ(1,1000) IORD(7,IP)
      IORD(6,IP)=ISMPLE(IORD(6,IP),FSAMP,0)
      IORD(4,IP)=ISMPLE(IORD(4,IP),FSAMP,0)
      IF(IORD(1,IP).EQ.12) GOTO 127
      IORD(5,IP)=1
      IORD(4,IP)=IORD(4,IP)-1
      GOTO 5
 127  IORD(5,IP)=ISMPLE(IORD(5,IP),FSAMP,0)
      IF(IORD(5,IP).LE.0) IDFLG=IDFLG+1
      GOTO 5
C
C  AUTOCORRELATION OUTPUT
C
 130  IAUTO=IAUTO+1
      IORD(2,IP)=IAUTO
```

```
      READ(1,1000) IORD(3,IP),IORD(4,IP)
      IF(IORD(3,IP).EQ.3) GOTO 134
      READ(1,1000) NAUBEG(IAUTO,1)
      GOTO 135
 134  READ(1,1000) (NAUBEG(IAUTO,I),I=1,NTRACO)
 135  IORD(4,IP)=ISMPLE(IORD(4,IP),FSAMP,0)
C
C  OPEN AUTOCORRELATION DISK FILE
C
      IORD(5,IP)=IORD(4,IP)/128
      IF(MOD(IORD(4,IP),128).NE.0) IORD(5,IP)=IORD(5,IP)+1
      IAUTOT=NFILES*IORD(5,IP)*NTRACO+1
      IF(IENTER(12+IAUTO,FSPECA(IAUTO),IAUTOT).LT.0)
     +STOP'DISK ENTER ERR'
      GOTO 5
C
C TRACE NORMALISATION
C
 140  READ(1,1000) IORD(2,IP)
      GOTO 5
C
C  HILBERT TRANSFORM
C
 150  GOTO 5
C
C  STACK
C
 160  READ(1,1000) IORD(2,IP),IORD(3,IP),IORD(5,IP)
      IF(IORD(2,IP).GT.0) IST=1
      IF(IORD(2,IP).GT.0) NTRACO=1
      IF(IORD(2,IP).EQ.2) GOTO 5
      ISFLG=1
      READ(1,1400) NGATHR,(XSTART(I),I=1,NGATHR),XSTEP
      IORD(4,IP)=NGATHR
 1400 FORMAT(I6,7F10.0)
      GOTO 5
C
C  THREE TRACE MIX
C
 170  ITM=IP
      NFILES=NFILES+1
      IORD(10,IP)=0
      READ(1,1100) R(1),R(2),R(3)
      RTOT=R(1)+R(2)+R(3)
      R(1)=R(1)/RTOT
      R(2)=R(2)/RTOT
      R(3)=R(3)/RTOT
      DO 172 I=1,6
 172  IORD(1+I,IP)=IR(I)
      GOTO 5
C
C  VELOCITY FILTER
C
 180  IVF=IP
      READ(1,1000) IORD(11,IP),IORD(12,IP),IORD(3,IP),IORD(5,IP)
      READ(1,1400) NGATHR,(XSTART(I),I=1,NGATHR),XSTEP
      IORD(4,IP)=NGATHR
      GOTO 5
C
C  DIP FILTER
C
 190  IDP=IP
      READ(1,1400) IORD(2,IP),XCMP,VAP
      FNQ=0.5*FSAMP
      SNQ=1.0/(2.0*XCMP)
 194  DO 196 I=1,2
      IORD(2+I,IP)=IVAP(I)
      IORD(4+I,IP)=IFNQ(I)
 196  IORD(6+I,IP)=ISNQ(I)
      GOTO 5
C
C  TIME VARYING BAND-PASS
```

```
C
 200  READ(1,1400) IORD(2,IP),FL1,FH1
      IORD(2,IP)=ISMPLE(IORD(2,IP),FSAMP,0)
      IF(IORD(2,IP).LT.0.0) IDFLG=IDLFG+1
      READ(1,1400) IORD(7,IP),FL2,FH2
      IORD(7,IP)=ISMPLE(IORD(7,IP),FSAMP,0)
      IF(IORD(7,IP).LT.0.0) IDFLG=IDLFG+1
      B1=ANGLE(FH1,FSAMP)-ANGLE(FL1,FSAMP)
      OM1=ANGLE(FH1,FSAMP)*ANGLE(FL1,FSAMP)
      B2=ANGLE(FH2,FSAMP)-ANGLE(FL2,FSAMP)
      OM2=ANGLE(FH2,FSAMP)*ANGLE(FL2,FSAMP)
      DO 202 I=1,2
      IORD(2+I,IP)=JB1(I)
      IORD(4+I,IP)=IOM1(I)
      IORD(7+I,IP)=JB2(I)
 202  IORD(9+I,IP)=IOM2(I)
      GOTO 5
C
C  HORIZONTAL STACKED TRACE SUMMING
C
 210  READ(1,1000) IORD(2,IP)
      GOTO 5
C
C  AUTO STATICS
C
 220  IAU=IP
      READ(1,1000) IORD(2,IP),IORD(3,IP)
      READ(1,1000) IORD(4,IP),IORD(5,IP),IORD(6,IP),IORD(7,IP)
      IORD(5,IP)=ISMPLE(IORD(5,IP),FSAMP,0)
      IORD(6,IP)=ISMPLE(IORD(6,IP),FSAMP,0)
      GOTO 5
C
C       EXIT CALL
C
 250  CONTINUE
C
C  GET ASCII INFORMATION TO GO IN REEL HEADER
C
      READ(1,1000) ILINE
      IF(ILINE.LE.0) GOTO 155
      BACKSPACE 1
      READ(1,3200) ILINE,((ITEXT(I,J),I=1,64),J=1,ILINE)
 3200 FORMAT(I6,(64A1))
 155  CLOSE(UNIT=1)
C
C  SET UP OFFSET ARRAY IF STACK OR VEL. FILTER USED
C
      IF(ISFLG.EQ.0.AND.IVF.EQ.0) GOTO 500
      DO 461 I=1,NGATHR
      XOFF=XSTART(I)
      DO 461 J=1,NTRACI
      X(J+(I-1)*NTRACI)=XOFF
 461  XOFF=XOFF+XSTEP
C
C  RETURN
C
 500  RETURN
      END


      FUNCTION ANGLE(FREQ,FSAMP)
C
C  GET ARC TANGENT
C
      ANGLE=ATAN(3.14158*FREQ/FSAMP)
      RETURN
      END
```

```
C  SET-UP SUBROUTINE FOR SEISMIC PROCESSING PACKAGE
C  ALLOCATES ARRAY PROCESSOR SPACE FOR DATA VECTORS
C  READS TAPE HEADERS FROM INPUT TAPE AND STORES THEM IN DK6:SEISHD.DAT
C  IF REQUIRED READS STACKING VELOCITY/TIME DATA FROM DK6:VELRMS.DAT
C              READS VELOCITY FILTER VELOCITY/TIME DATA FROM DK6:VELFIL.DAT
C              READS DEPTH DATA FROM DK6:DEPTH.DAT IF PRESENT
C
C              OPENS DISK FILE DR6:VFSEIS.DAT FOR VELOCITY FILTER AND
C                                                 AUTOSTATICS
C              OPENS DISK FILE DR6:DPSEIS.DAT FOR DIP FILTER
C
C  OPENS LOG FILE DK6:SEIS.LOG
C
      SUBROUTINE SEISIO(SEIS,IDEPTH,V,VF,RLHDBL,RLDUM)
C
      VIRTUAL RLHDBL(80,40),RLDUM(200),IDEPTH(10000),V(2,11,100),
     +VF(2,11,100),SEIS(4156)
      REAL*8 FSPECA,FSPECB
      REAL*4 TMNAM(3)
      INTEGER*2 IORD(12,20),RLDUM,REELNU,TPDRR,TPDRW,ITMERR(2),
     +LOC(20)
      LOGICAL*1 RLHDBL
C
      COMMON/INPUT/ NFILES,NCHAN,NSMPIN,ISTART,ILEN,IFIRST,ILAST,
     +FTRCNO,IWORDR,IWORDW,IBLK,IVFRT,IDPRT,NTRACI
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/CTRL/  ITM,IST,IVF,IDP,IAU,IORD
      COMMON/STATDP/IDFLG,IDPTOT,IDPINT,ISSHOT,IXES,IXSR,IXSS,IXRR
      COMMON/REELS/ ILINE,REELNU,NTRACO
      COMMON/TAPE/  TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      DATA FSPECA/12RDR6VFSEISDAT/,FSPECB/12RDR6DPSEISDAT/
C
C  SET UP CONSTANTS
C
      IWORDR=2*NSMPIN+120
      IWORDW=2*NSAMP+120
      IBLK=IWORDW/256
      IF(MOD(IWORDW,256).NE.0) IBLK=IBLK+1
      IF(IVF.EQ.0.AND.IAU.EQ.0) GOTO 350
      IVFRT=IGETC()
      IF(IENTER(IVFRT,FSPECA,NTRACI*IBLK).LT.0)
     +STOP'DR DISK DATA FILE ENTER ERROR'
 350  IEOTW=0
      IA=0
      IB=IA+NSAMP
      IC=IB+NSAMP
      ID=IC+NSAMP
      IG=24576-NSAMP
      IE=IG-NSAMP
      IF(REELNU.EQ.0) REELNU=1
      IF(FTRCNO.EQ.0.0) FTRCNO=1.0
C
C  LOOKUP READ TAPE DRIVE
C
 370  ICHN=TPDRR+10
      ILK=LOOKUP(ICHN,TMNAM(TPDRR+1),-1)
      ISTUS=IPEEK("172520)
      IF((ISTUS.AND."100).GT.0) GOTO 380
      CALL CLOSEC(ICHN)
      TYPE 3000
 3000 FORMAT(' GIVEN READ DRIVE NOT AVAILABLE',/,
     +'$GIVE NEW READ DRIVE NO.:')
      ACCEPT 3100,TPDRR
 3100 FORMAT(I2)
      IF(TPDRR.GE.3) STOP'NON-EXISTANT READ DRIVE'
      GOTO 370
C
C  IF LOOKUP OK, CHECK BOT FLAG
C
 380  IEOTR=0
```

```
      IF((ISTUS.AND."40).LE.0) GOTO 410
C
C  READ REEL HEADERS FROM TAPE
C
      IF(IRMTXM(RLHDBL(1,1),ICHN,1600,ITMERR(1)).NE.1600) STOP'ERROR
     + READING ASCII TAPE HEADER'
      CALL ISPFNW("376,ICHN)
      IF(IRMTXM(RLDUM(1),ICHN,200,ITMERR(1)).NE.200) STOP'ERROR
     + READING BINARY TAPE HEADER'
      CALL ISPFNW("376,ICHN)
C
C  OPEN DISC FILE FOR PERMANENT STORAGE OF TAPE HEADER BLOCKS
C
      OPEN(UNIT=3,NAME='DK6:SEISHD.DAT',TYPE='NEW',ERR=390)
      GOTO 400
 390  STOP'OPEN DK6:SEISHD.DAT ERROR'
 400  WRITE(3,4100)((RLHDBL(I,J),I=1,80),J=1,40)
 4100 FORMAT(' ',80A1)
      WRITE(3,4200)(RLDUM(I),I=1,200)
 4200 FORMAT(' ',8O6)
      CLOSE(UNIT=3)
C
C  GET VELOCITY DATA FROM FILE
C
 410  IP=0
      NUM=0
 161  IP=IP+1
      IF(IORD(1,IP).EQ.25) GOTO 162
      IF(IORD(1,IP).NE.16) GOTO 161
      IF(IORD(2,IP).EQ.2) GOTO 161
      NUM=NUM+1
      LOC(NUM)=IP
      GOTO 161
 162  IF(NUM.EQ.0) GOTO 440
      ICNT=0
      OPEN(UNIT=2,NAME='DK6:VELRMS.DAT',TYPE='OLD',ERR=164)
      GOTO 165
 164  STOP'NO DK6:VELRMS.DAT FILE'
 165  CONTINUE
      READ(2,1000,END=170) NCDP,NLYR
 1000 FORMAT(1216)
      ICNT=ICNT+1
      V(1,1,ICNT)=FLOAT(NCDP)
      V(2,1,ICNT)=FLOAT(NLYR)
      DO 166 J=1,NLYR
      READ(2,1100)V(1,J+1,ICNT),V(2,J+1,ICNT)
 1100 FORMAT(2F10.0)
 166  CONTINUE
      GOTO 165
 170  DO 175 I=1,NUM
      IORD(9,LOC(I))=1
 175  IORD(6,LOC(I))=ICNT
      CLOSE(UNIT=2)
C
C  GET VELOCITY FILTER VELOCITIES FROM FILE
C
 440  IF(IVF.EQ.0) GOTO 460
      OPEN(UNIT=2,NAME='DK6:VELFIL.DAT',TYPE='OLD',ERR=154)
      GOTO 155
 154  STOP'NO DK6:VELFIL.DAT FILE'
 155  CONTINUE
      READ(2,1000,END=160) NCDP,NLYR
      IORD(6,IVF)=IORD(6,IVF)+1
      VF(1,1,IORD(6,IVF))=FLOAT(NCDP)
      VF(2,1,IORD(6,IVF))=FLOAT(NLYR)
      DO 156 J=1,NLYR
      READ(2,1100)VF(1,J+1,IORD(6,IVF)),VF(2,J+1,IORD(6,IVF))
 156  CONTINUE
      GOTO 155
 160  IORD(9,IVF)=1
      CLOSE(UNIT=2)
C
```

```
C  OPEN FILE FOR DIP FILTER
C
 460  IF(IDP.EQ.0) GOTO 480
      IORD(10,IDP)=0
      IORD(9,IDP)=IORD(2,IDP)/2
      IDPRT=IGETC()
      IF(IENTER(IDPRT,FSPECB,IORD(2,IDP)*IBLK).LT.0)
     +STOP'DIP FILTER DISK FILE ENTER ERROR'
      CALL APWAIT
      CALL VCLR(IA,1,NSAMP)
      CALL APWR
      CALL APGTXM(SEIS(61),NSAMP,IA,2)
      CALL APWD
      TYPE 5555,(IORD(III,IDP),III=1,12)
 5555 FORMAT(' ',12I5)
      TYPE 5555,IBLK,NSAMP,IWORDW
      DO 178 I=1,IORD(2,IDP)
      TYPE 5555,(I-1)*IBLK
      IWT=IWDRXM(SEIS(1),IDPRT,IWORDW,(I-1)*IBLK,0)
      IF(IWT.NE.IWORDW) STOP'DIP FILTER FILE CLEAR ERROR'
 178  CONTINUE
      NFILES=NFILES+IORD(9,IDP)
C
C  GET DEPTH DATA
C
 480  IF(IDFLG.EQ.0) GOTO 500
      OPEN(UNIT=4,TYPE='OLD',NAME='DK6:DEPTH.DAT',ERR=450)
      READ(4,1000) IYEAR,IDAY,IHOUR,IMIN,ISEC
      READ(4,1000) ISSINT,IXES,IXSR,IXSS,IXRR
      READ(4,1000) ISSHOT
      READ(4,1000) IDPINT
      READ(4,1000) IDPTOT
      IF(IDPTOT.LE.1) STOP 'NOT ENOUGH DEPTH DATA'
      READ(4,1200) (IDEPTH(I),I=1,IDPTOT)
 1200 FORMAT(I6)
      CLOSE(UNIT=4)
      IDPINT=IDPINT/ISSINT
      IDFLG=1
      GOTO 500
 450  TYPE 2500
 2500 FORMAT(/,' * WARNING * NO DEPTH DATA FILE DK6:DEPTH.DAT',/,
     +' ASSUMING STACKED DATA WITH A VELOCITY TABLE IN TRACE HEADERS',/,
     +' AND WILL USE FIRST TWO-WAY TRAVEL TIME',/)
       IDFLG=-1
C
C  OPEN LOG FILE DK6:SEIS.LOG
C
 500  OPEN(UNIT=2,NAME='DK6:SEIS.LOG',TYPE='NEW',ERR=420)
      GOTO 430
 420  STOP'LOG FILE OPEN ERR'
 430  WRITE(2,4310)
 4310 FORMAT(//,' SEISMIC PROCESSING LOG FILE',//)
      RETURN
      END
```

```
C  SUBROUTINE TO WRITE TAPE HEADERS TO OUTPUT TAPE
C  READS DATA FROM DISK FILE DK6:SEISHD.DAT AND UP-DATES IT
C  INSERTS ASCII TEXT AND TODAYS DATE INTO NEXT AVAILABLE BLOCK
C  AFTER LINE 25
C
      SUBROUTINE SEISHD(ITEXT,ICDPNO,ICALL,ISTOP)
C
      VIRTUAL ITEXT(64,10)
      REAL*4 CLRBF1(800),TMNAM(3)
      LOGICAL*1 RLHDBL(80,40),CLIENT(24),LINE(12),AREA(23),
     %REELNO(11),DAY(4),YEAR(5),OBSVER(17),TRINRC(8),CDPFLD(10),
     %SMPINT(9),SPINTR(7),BITS(6),BYTES(7),RECFMT(8),RLFMT(8),
     %UNITS(10),SFLTPT(5),GFLTPT(5),SOURCE(12),SORCNO(8),SCPTIN(18),
     %SWIDTH(18),SRCPTN(27),SPDOFS(8),SPDDIS(8),GRPINT(20),GEOGPN(5),
     %RLHDCD(400),TODAY(10),TDAY(2),TMON(2),TYEAR(4),ITEXT
      INTEGER*2 REELNU,TRCNUM,AUXTRS,SAMPIN,ORSINT,NOSMPL,ORSAMP,
     %FMTCD,CDPNO,SORTCD,VERTSM,BINGAN,AMPREC,UNITNO,POLARY,
     +TPDRR,TPDRW,ITMERR(2),RLDUM(200)
      INTEGER*4 LNNOL,RLNOL
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/REELS/ ILINE,REELNU,NTRACO
      COMMON/TAPE/  TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
C
      EQUIVALENCE (RLHDBL(1,1),CLRBF1(1)),(RLHDBL(1,1),RLHDCD(1)),
     +(RLHDCD(1),RLDUM(1))
      EQUIVALENCE (RLHDBL(11,1),CLIENT(1)),(RLHDBL(9,2),LINE(1)),
     %(RLHDBL(26,2),AREA(1)),(RLHDBL(12,3),REELNO(1)),
     %(RLHDBL(41,3),DAY(1)),(RLHDBL(50,3),YEAR(1)),
     %(RLHDBL(64,3),OBSVER(1)),(RLHDBL(23,5),TRINRC(1)),
     %(RLHDBL(71,5),CDPFLD(1)),(RLHDBL(20,6),SMPINT(1)),
     %(RLHDBL(42,6),SPINTR(1)),(RLHDBL(56,6),BITS(1)),
     %(RLHDBL(74,6),BYTES(1)),(RLHDBL(21,7),RECFMT(1)),
     %(RLHDBL(45,7),RLFMT(1)),(RLHDBL(71,7),UNITS(1)),
     %(RLHDBL(29,8),SFLTPT(1)),(RLHDBL(52,9),GFLTPT(1)),
     %(RLHDBL(17,11),SOURCE(1)),(RLHDBL(41,11),SORCNO(1)),
     %(RLHDBL(63,11),SCPTIN(1)),(RLHDBL(63,12),SWIDTH(1)),
     $(RLHDBL(17,12),SRCPTN(1)),(RLHDBL(19,15),SPDOFS(1)),
     $(RLHDBL(39,15),SPDDIS(1)),(RLHDBL(61,15),GRPINT(1)),
     $(RLHDBL(25,16),GEOGPN(1))
      EQUIVALENCE (RLHDCD(5),LNNOL),(RLHDCD(9),RLNOL),
     %(RLHDCD(13),TRCNUM),(RLHDCD(15),AUXTRS),(RLHDCD(17),SAMPIN),
     %(RLHDCD(19),ORSINT),(RLHDCD(21),NOSMPL),(RLHDCD(23),ORSAMP),
     %(RLHDCD(25),FMTCD),(RLHDCD(27),CDPNO),(RLHDCD(29),SORTCD),
     %(RLHDCD(31),VERTSM),(RLHDCD(51),BINGAN),(RLHDCD(53),AMPREC),
     %(RLHDCD(55),UNITNO),(RLHDCD(57),POLARY)
      EQUIVALENCE (TODAY(1),TDAY(1)),(TODAY(4),TMON(1)),
     +(TODAY(7),TYEAR(1))
C
C     ASSIGN CHANNEL NUMBERS
C
 10   ICHN=TPDRW+10
C
C     LOOKUP TAPE DRIVE
C
      TYPE 2000,TPDRW,ICDPNO
      WRITE(2,2000) TPDRW,ICDPNO
 2000 FORMAT(' START NEW WRITE DRIVE:',I1,' CPD NUMBER:'I5,//)
      ILK=LOOKUP(ICHN,TMNAM(TPDRW+1),-1)
      ISTUS=IPEEK("172520)
      IF((ISTUS.AND."100).GT.0) GOTO 20
      TYPE 2100,TPDRW,ICDPNO
 2100 FORMAT(' GIVEN DRIVE NO:'I1,' NOT AVAILABLE AT CDP NO:',I5)
 15   CALL CLOSEC(ICHN)
      TYPE 2150
 2150 FORMAT('$GIVE NEW WRITE DRIVE NO.:')
      ACCEPT 1000,TPDRW
 1000 FORMAT(I1)
      IF(TPDRW.LE.2) GO TO 10
      ISTOP=1
      RETURN
C
```

```
C     CHECK WRITE DRIVE IS NOT WRITE PROTECTED
C
 20   IF((ISTUS.AND."4).EQ.0) GOTO 30
      TYPE 2300,TPDRW
 2300 FORMAT(' GIVEN WRITE DRIVE NO:',I1,' IS WRITE PROTECTED')
      GOTO 15
C
C  READ TAPE HEADER BLOCKS FROM DISK
C
 30   OPEN(UNIT=3,NAME='DK6:SEISHD.DAT',TYPE='OLD',ERR=40)
      IHD=1
      GO TO 50
 40   TYPE 2400
 2400 FORMAT(' NO TAPE HEADERS ON DISK')
      IHD=0
 50   DO 42 I=1,800
 42   CLRBF1(I)=0.0
      IF(IHD.EQ.1) READ(3,1100)((RLHDBL(I,J),I=1,80),J=1,40)
 1100 FORMAT(' ',80A1)
C
C  GET DATE
C
      CALL DATE(IDAY,IMONTH,IYEAR)
      ENCODE(2,1400,TDAY,ERR=70) IDAY
      ENCODE(2,1400,TMON,ERR=70) IMONTH
      ENCODE(4,1400,TYEAR,ERR=70) IYEAR
      TODAY(3)="57
      TODAY(6)="57
C
C  UPDATE REEL HEADERS
C
      ENCODE(11,1300,REELNO,ERR=70) REELNU
      ENCODE(8,1400,TRINRC,ERR=70) NTRACO
      ENCODE(7,1400,SPINTR,ERR=70) NSAMP
 1300 FORMAT(I7)
 1400 FORMAT(I6)
      IF(ILINE.EQ.0) GOTO 80
      ICHR=0
      DO 62 I=25,39
      NLINE=I
      DO 60 J=5,80
      IF(RLHDBL(J,I).NE.' ') GOTO 63
 60   CONTINUE
      ICHR=ICHR+1
      IF(ICHR.EQ.2) GOTO 64
      GOTO 62
 63   ICHR=0
 62   CONTINUE
 64   IF(NLINE+ILINE.LE.40) GOTO 66
      TYPE 2500
 2500 FORMAT(' * WARNING * NO ROOM FOR ASCII IN REEL HEADER')
      GOTO 80
 66   NLINE=NLINE-1
      DO 67 J=1,10
 67   RLHDBL(4+J,NLINE+1)=TODAY(J)
      DO 68 I=1,ILINE
      DO 68 J=1,64
 68   RLHDBL(15+J,NLINE+I)=ITEXT(J,I)
      GO TO 80
 70   ISTOP=2
      RETURN
C
C  IF FIRST CALL TO SEISHD FROM ROOT SEGMENT WRITE TAPE HEADER TO LOG FILE
C
 80   IF(ICALL.EQ.1) WRITE(2,1100)
     +((RLHDBL(I,J),I=1,80),J=1,40)
      IF(ICALL.EQ.1) WRITE(2,1500)
 1500 FORMAT(///)
C
C  IF TAPE NOT AT BOT RETURN
C
      IF((ISTUS.AND."40).EQ.0) GOTO 200
```

```
      IWT=IWMTLM(RLHDBL(1,1),ICHN,1600,ITMERR(1))
      IF(IWT.EQ.1600) GOTO 90
      ISTOP=3
      RETURN
90    CALL ISPFNW("377,ICHN)
      DO 92 I=1,100
92    CLRBF1(I)=0.0
      IF(IHD.EQ.1) READ(3,1200)(RLDUM(I),I=1,200)
1200  FORMAT(' ',8O6)
      CALL JICVT(REELNU,RLNOL)
      CALL JJCVT(RLNOL)
      TRCNUM=NTRACO
      NOSMPL=NSAMP
      IWT=IWMTLM(RLDUM,ICHN,200,ITMERR(1))
      IF(IWT.EQ.200) GOTO 100
      ISTOP=4
      RETURN
100   CALL ISPFNW("377,ICHN)
200   IF(IHD.EQ.1) CLOSE(UNIT=3)
      CALL IWAIT(ICHN)
      RETURN
      END



C  SEISMIC PROCESSING CALLING SUBROUTINE.
C  PROCESSING OPTIONS ARE CALLED UNDER THE CONTROL OF THE DATA
C  ARRAY IORD(X,Y)
C
      SUBROUTINE SEISSB(SEIS,ICDPED,ITRCED,ITRACP,IDEPTH,
     +NBEG,NAUBEG,V,VF,X,ITEXT,ICDPNO,ISTOP)
C
      VIRTUAL ICDPED(100),ITRCED(100),ITRACP(48),IDEPTH(10000),
     +ITEXT(64,10),FILTER(4096),SEIS(4156),NBEG(3,96),NAUBEG(2,96),
     +V(2,11,100),X(96),DIP(1024),T0LYR(10),VLYR(10),SEISA(4156),
     +SEISB(4156),VF(2,11,100),AUTO(4096),XM(22),TRHD(96,60),
     +RESULT(32),SSUM(4096)
      REAL*4 TRHDBF(60),TMNAM(3)
      INTEGER*4 JCDPNO,JCDPTR
      INTEGER*2 IORD(12,20),REELNU,TPDRR,TPDRW,ITMERR(2),IT0LYR(10),
     +IVLYR(10),IRTBLK(2),ITA(2)
      LOGICAL*1 OUTBUF(240),ITEXT,IANS,LDUM
C
      COMMON/INPUT/ NFILES,NCHAN,NSMPIN,ISTART,ILEN,IFIRST,ILAST,
     +FTRCNO,IWORDR,IWORDW,IBLK,IVFRT,IDPRT,NTRACI
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/CTRL/  ITM,IST,IVF,IDP,IAU,IORD
      COMMON/STATDP/IDFLG,IDPTOT,IDPINT,ISSHOT,IXES,IXSR,IXSS,IXRR
      COMMON/DATBUF/OUTBUF
      COMMON/REELS/ ILINE,REELNU,NTRACO
      COMMON/TAPE/  TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(21),JCDPNO),(OUTBUF(25),JCDPTR),
     +(OUTBUF(29),ITYPE),(OUTBUF(31),IFOLD),(OUTBUF(109),IDELAY),
     +(OUTBUF(111),MUTST),(OUTBUF(113),MUTEND),(OUTBUF(115),NSMP),
     +(OUTBUF(199),ILYR),(OUTBUF(201),IT0LYR(1)),
     +(OUTBUF(221),IVLYR(1))
      EQUIVALENCE (OUTBUF(1),TRHDBF(1))
C
C  START MAIN PROCESSING LOOP
C
      IF(IDP.NE.0) CALL OPERDP(DIP,IORD(1,IDP))
      IRTBLK(1)=1
      IRTBLK(2)=1
      DO 999 IFILES=1,NFILES
      IFIL=IFILES
 40   IPS=0
      IDCDP=0
      ITA(1)=0
```

```
      ITA(2)=0
      ITRI=0
      ITRO=0
      LCHAN=NCHAN
 50   IVF=0
      ITVF=0
      ITDR=0
      IAST=0
      IAS=0
      CALL APWAIT
      CALL ZMD
      CALL APWR
C
C  ITERATE THROUGH INPUT CHANNELS
C
      DO 800 ICHAN=1,LCHAN
      ICHA=ICHAN
      CALL VCLR(IA,1,NSAMP)
      IF(ITM.EQ.0.OR.IFIL.LT.NFILES) GOTO 60
      IF(ICHA.LT.NCHAN) GOTO 800
      IPS=ITM-1
      GOTO 380
 60   IF(IDP.EQ.0.OR.IFIL.LE.NFILES-IORD(9,IDP)) GOTO 61
      IF(ICHA.LT.NCHAN) GOTO 800
      IPS=IDP-1
      GOTO 380
 61   CALL APWAIT
C
C  GET NEXT TRACE
C
      CALL TRINPT(SEIS,IDEPTH,IPS,ITDR,ITRI,ITRO,ICDPNO,
     +ICHA,ITRIN,IDCDP,IDS,IDR,ISTOP)
      IF(ISTOP.NE.0) RETURN
      IF(ITRIN.EQ.-1) GOTO 562
C
C  DO REST OF PROCESSING AS REQUIRED
C
 380  IP=IPS
 400  IP=IP+1
      CALL APWAIT
D      CALL APGTXM(SEIS(61),0,50,2)
D      CALL APWAIT
D      WRITE(2,5555) (SEIS(60+I),I=1,50)
D5555 FORMAT(' ',10E12.5)
D      TYPE *,(SEIS(60+I),I=1,50)
D      TYPE *,ITRI,ITRO,ICDPNO,IORD(1,IP)
D      TYPE 555,(IORD(I,IP),I=1,12)
D 555  FORMAT(' ',618)
      GOTO(410,420,430,440,450,460,470,480,490,500,510,
     +520,530,540,550,560,570,580,590,600,
     +610,620,650,650,650)IORD(1,IP)
C
C  APPLY TIME BREAK AND SOURCE/RECEIVER STATIC CORRECTION
C
 410  CALL STATIC(IORD(1,IP))
      GOTO 400
C
C PROCESS EDIT COMMANDS
C
 420  CALL EDIT(SEIS,ICDPED,ITRCED,ICDPNO,ITRIN,IORD(1,IP))
      GOTO 400
C
C  MUTE APPLICATION
C
 430  CALL TMUTE(IDCDP,ITRIN,IORD(1,IP))
      GOTO 400
C
C POLARITY REVERSAL
C
 440  DO 431 IPOL=1,IORD(2,IP)
      IF(ITRACP(IPOL).EQ.ITRIN)GOTO 435
 431  CONTINUE
```

```
      GOTO 400
 435  CALL VNEG(IA,1,IA,1,NSAMP)
      CALL APWAIT
      GOTO 400
C
C  SEA-FLOOR CONSISTANT STATICS
C
 450  ITDEL=IDS+IDR-2*IDCDP
      CALL TMOV(ITDEL)
      GOTO 400
C
C AMP RECOVERY FILTERS APPLICATION AND REMOVAL
C
 460  CALL TRAMP(IORD(1,IP))
      GOTO 400
C
C TV**2 FILTER
C
 470  CALL TVRAMP(IORD(1,IP))
      GOTO 400
C
C  AGC
C
 480  CALL AGC(IDCDP,IORD(1,IP))
      GOTO 400
C
C BANDPASS FILTER
C
 490  CALL BANDPS(IORD(1,IP))
      GOTO 400
C
C BANDREJECT FILTER
C
 500  CALL BANDRJ(IORD(1,IP))
      GOTO 400
C
C WIENER SPIKING FILTER AND
C PREDICTION ERROR FILTER
C
 510  CONTINUE
 520  IF(ITRIN.EQ.1) IORD(10,IP)=0
      IF(ITYPE.NE.1) GOTO 400
      CALL DESIGN(FILTER,NBEG,LAG,ICDPNO,ITRIN,IDCDP,IORD(1,IP))
      IF(IORD(9,IP).EQ.0) CALL APPLY(FILTER,IORD(1,IP))
      GOTO 400
C
C  AUTOCORRELATION OUTPUT
C
 530  CALL AUTOUT(AUTO,NAUBEG,NTRACI,ITRIN,IRTBLK,ITA,IORD(1,IP),
     +ISTOP)
      IF(ISTOP.NE.0) RETURN
      GOTO 400
C
C NORMALISATION
C
 540  CALL APWAIT
      IF(IORD(2,IP).GE.0)GOTO 545
      CALL MAXMGV(IA,1,ID,NSAMP)
      GOTO 546
 545  CALL SVESQ(IA,1,ID,NSAMP)
      IF(IORD(2,IP).EQ.0) CALL VSQRT(ID,1,ID,1,1)
 546  CALL VSDIV(IA,1,ID,IA,1,NSAMP)
      CALL APWAIT
      GOTO 400
C
C  GET REFLECTION STRENGTH COEFFICIENTS
C
 550  CALL HILBRT
      GOTO 400
C
C  STACK
C
```

```
560   IF(IFIL.EQ.1) NOFFST=MOD(ICDPNO,IORD(4,IP))
      IORD(8,IP)=MOD((ICDPNO-NOFFST),IORD(4,IP))
      IF(ITRIN.EQ.1)
     +CALL STKSET(V,T0LYR,VLYR,ICDPNO,IORD(1,IP))
      IF(ITYPE.NE.1) GOTO 562
      CALL STACK(T0LYR,VLYR,X,XM,NTRACI,ITRIN,IORD(1,IP))
562   IF(IP.EQ.0) GOTO 565
      IF(IORD(2,IP).EQ.2) GOTO 565
      ILYR=IORD(7,IP)
      DO 567 I=1,ILYR
      IT0LYR(I)=IFIX(T0LYR(I))
567   IVLYR(I)=IFIX(1000.0*VLYR(I))
      IF(ITRIN.EQ.-1) GOTO 565
      IF(IORD(2,IP).LT.0) GOTO 400
565   IF(ICHA.LT.NCHAN.OR.IST.EQ.0) GOTO 800
      IF(IDFLG.LT.0) IDCDP=ISMPLE(IT0LYR(1),FSAMP*0.5,0)
      IFOLD=IORD(10,IP)
      ITRIN=1
      CALL JICVT(1,JCDPTR)
      CALL JJCVT(JCDPTR)
      ITYPE=1
      IF(IFOLD.EQ.0) ITYPE=2
      CALL APWAIT
      CALL LVNOT(IG,1,IB,1,NSAMP)
      CALL VADD(IG,1,IB,1,IB,1,NSAMP)
      CALL VDIV(IB,1,IE,1,IA,1,NSAMP)
      CALL APWR
      GOTO 400
C
C  THREE TRACE MIX
C
570   CALL TMIX(SEISA,SEISB,ICDPNO,IORD(1,IP))
      IF(IORD(10,IP).EQ.1) GOTO 800
      GOTO 400
C
C  VELOCITY FILTER OUTPUT TO DISK FILE
C
580   ITVF=ITVF+1
      IVF=IP
      NBLK=(ITVF-1)*IBLK
      CALL DISKIN(SEIS,IVFRT,IWORDW,NBLK,ISTOP)
      IF(ISTOP.NE.0) RETURN
      GOTO 800
C
C  DIP FILTER
C
590   IORD(10,IP)=IORD(10,IP)+1
      NBLK=MOD((IORD(10,IP)+IORD(9,IP)),IORD(2,IP))*IBLK
      CALL DISKIN(SEIS,IDPRT,IWORDW,NBLK,ISTOP)
      IF(ISTOP.NE.0) RETURN
      IF(IORD(10,IP).LE.IORD(9,IP)) GOTO 800
      CALL DIPFIL(SEIS,DIP,ICDPNO,IORD(1,IP),ISTOP)
      IF(ISTOP.NE.0) RETURN
      GOTO 400
C
C  TIME VARYING BAND-PASS
C
600   CALL TVBAND(IDCDP,IORD(1,IP))
      GOTO 400
C
C   HORIZONTAL STACKED TRACE SUMMING
C
610   CALL HSUM(SSUM,IORD(1,IP))
      IF(IORD(3,IP).EQ.1) GOTO 400
      GOTO 800
C
C  AUTO STATICS
C
620   IAST=IAST+1
      IAS=IP
      NBLK=(IAST-1)*IBLK
      CALL DISKIN(SEIS,IVFRT,IWORDW,NBLK,ISTOP)
```

```
      IF(ISTOP.NE.0) RETURN
      GOTO 800
C
C  EXIT CALL
C
 650  ITRO=ITRO+1
      IF(ITYPE.EQ.1) GOTO 750
      WRITE(2,2000) ICDPNO,ITRO
 2000 FORMAT(' CDP NO:',I5,' O/P TRACE NO:',I2,' BLANKED')
C
C END OF PROCESS LOOP
C
 750  CALL APWAIT
      CALL APGTXM(SEIS(61),IA,NSAMP,2)
      CALL APWAIT
 780  CALL TRWRIT(SEIS,ICDPNO,IWORDW,FTRCNO,ITRI,ITRO,ISTOP)
      IF(ISTOP.NE.0) RETURN
      IF(IEOTW.EQ.0) GOTO 800
      IRTBLK(1)=IRTBLK(1)-ITA(1)
      IRTBLK(2)=IRTBLK(2)-ITA(2)
      IEOTW=0
      REELNU=REELNU+1
      CALL SEISHD(ITEXT,ICDPNO,0,ISTOP)
      IF(ISTOP.NE.0) RETURN
      GOTO 40
C
C  END OF TRACE LOOP
C
 800  CONTINUE
C
C  CHECK FOR VELOCITY FILTER
C
      IF(IVF.EQ.0) GOTO 870
      IF(IFIL.EQ.1) NOFFST=MOD(ICDPNO,IORD(4,IVF))
      IORD(8,IVF)=MOD((ICDPNO-NOFFST),IORD(4,IVF))
      CALL STKSET(VF,T0LYR,VLYR,ICDPNO,IORD(1,IVF))
      CALL VELFIL(SEIS,T0LYR,VLYR,X,XM,TRHD,RESULT,ICDPNO,
     +IORD(1,IVF),ISTOP)
      IF(ISTOP.NE.0) RETURN
      IPS=IVF
      LCHAN=NTRACI
      GOTO 50
C
C  CHECK FOR AUTO STATICS
C
 870  IF(IAS.EQ.0) GOTO 880
      CALL AUTOST(SEIS,ICDPNO,IORD(1,IAS),ISTOP)
      IF(ISTOP.NE.0) RETURN
      IPS=IAS
      LCHAN=NTRACI
      GOTO 50
C
C END OF FILE LOOP
C
 880  IF(MOD(IFIL,10).EQ.0) TYPE 9000,IFIL,ICDPNO
 9000 FORMAT('+NO. OF FILES PROCESSED:',I5,' CDP NO.:'I5)
 999  CONTINUE
C
C  RETURN
C
      RETURN
      END
```

```
      SUBROUTINE DISKIN(SEIS,IRT,IWORD,IBLK,ISTOP)
C
      VIRTUAL SEIS(4156)
      REAL*4 TRHDBF(60)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
C
      EQUIVALENCE (OUTBUF(1),TRHDBF(1))
C
      DO 100 I=1,60
 100  SEIS(I)=TRHDBF(I)
      CALL APWAIT
      CALL APGTXM(SEIS(61),IA,NSAMP,2)
      CALL APWAIT
      IWT=IWDRXM(SEIS(1),IRT,IWORD,IBLK,0)
      IF(IWT.NE.IWORD) ISTOP=34
      RETURN
      END



      SUBROUTINE TMOV(ITB)
C
C  SUBROUTINE TO ITB TRACE RELATIVE TO START POINT
C  BY ITB SAMPLES.
C  ITB>0 MOVES TRACE UP, I.E. INCREASES DELAY
C  ITB<0 MOVES TRACE DOWN, I.E. DECREASES DELAY
C
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      JTB=IABS(ITB)
      IF(JTB.GT.NSAMP) STOP'TMOV ERR'
      IF(ITB.LE.0) GOTO 411
      CALL VCLR(IB,1,NSAMP)
      CALL VMOV(JTB,1,IA,1,NSAMP)
      CALL APWR
 411  IF(ITB.GE.0) GOTO 412
      CALL VMOV(IA,1,IB,1,NSAMP)
      CALL VCLR(IA,1,NSAMP)
      CALL VMOV(IB,1,JTB,1,NSAMP)
      CALL APWR
 412  CONTINUE
      CALL APWAIT
      RETURN
      END



      SUBROUTINE STATIC(IORD)
C
      INTEGER*4 JORFTR,JRECDP,JSORDP
      INTEGER*2 IORD(12)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(13),JORFTR),(OUTBUF(41),JRECDP),
     +(OUTBUF(49),JSORDP),(OUTBUF(99),ISST),(OUTBUF(101),IRST),
     +(OUTBUF(103),ITOTST)
C
      CALL JJCVT(JORFTR)
```

```
      CALL IJCVT(JORFTR,IORFTR)
      CALL JJCVT(JORFTR)
      CALL JJCVT(JRECDP)
      CALL IJCVT(JRECDP,IRECDP)
      CALL JJCVT(JRECDP)
      CALL JJCVT(JSORDP)
      CALL IJCVT(JSORDP,ISORDP)
      CALL JJCVT(JSORDP)
C
C  SEE IF SOURCE DEPTH TO BE TAKEN FROM HEADER
C
      ISST=IORD(3)
      IF(IORD(3).LT.0) ISST=ISORDP
      IRST=IORD(4)
      IF(IORD(4).GT.0) GOTO 100
      IF(IORD(4).LT.0) IRST=-IRECDP
      GOTO 200
C
C  GET RECEIVER DEPTHS FROM I/P DATA TABLE
C
 100  DO 150 I=1,IORD(4)
      J=2*I
      K=2*(I-1)
      IF(IORFTR.LT.IORD(J+3)) GOTO 160
 150  CONTINUE
      GOTO 170
 160  IF(K.EQ.0) GOTO 170
      IRST=IORD(K+4)+IFIX(FLOAT(IORD(J+4)-IORD(K+4))*
     +FLOAT(IORFTR-IORD(K+3))/FLOAT(IORD(J+3)-IORD(K+3)))
      GOTO 200
 170  IRST=IORD(J+4)
C
C  GET TOTAL STATIC AND ADD TIME BREAK CORRECTION
C
 200  ITOTST=ITOTST+ISST+IRST
      ICORR=ISMPLE(IORD(2),FSAMP,ISMPLE(ITOTST,0.667,0))
      CALL TMOV(ICORR)
C
C  RETURN
C
      RETURN
      END




      SUBROUTINE EDIT(SEIS,ICDPED,ITRCED,ICDPNO,ICDPTR,IORD)
C
      VIRTUAL SEIS(4156),ICDPED(100),ITRCED(100)
      INTEGER*2 IORD(12),IEDLEV(2)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (IEDLEV(1),EDLEV),(OUTBUF(29),ITYPE)
C
      IF(IORD(9).GT.IORD(2)) GOTO 10
      IF(ICDPED(IORD(9)).NE.ICDPNO)GOTO 10
      IF(ITRCED(IORD(9)).NE.ICDPTR)GOTO 10
      IORD(9)=IORD(9)+1
      GOTO 40
  10  IF(IORD(3).EQ.0) RETURN
      DO 15 I=1,2
  15  IEDLEV(I)=IORD(6+I)
      ISUM=0
      CALL APPUT(EDLEV,ID,1,2)
      CALL APWD
C
C  GET WINDOW FOR SEARCH AND FLAG ALL VALUES ABOVE EDLEV
```

```
C
      IEDST=MAX0(0,ISMPLE(IORD(4),FSAMP,IDELAY))
      CALL VABS(IEDST,1,IB,1,IORD(5))
      CALL LVGT(IB,1,ID,0,IC,1,IORD(5))
      CALL APWR
C
C  IF ZEROING TRACE JUMP TO NEXT SECTION OTHERWISE REMOVE SPIKES BY
C  SETTING TO ZERO
C
      IF(IORD(6).GT.0) GOTO 18
      CALL LVNOT(IC,1,IB,1,IORD(5))
      CALL VMUL(IEDST,1,IB,1,IEDST,1,IORD(5))
      CALL APWR
      RETURN
C
C  GET FLAGGED SPIKE ARRAY AND CHECK IF TO BE ZEROED
C
 18   CALL APGTXM(SEIS(61),IC,IORD(5),2)
      CALL APWD
      DO 30 I=1,IORD(5)
      IF(SEIS(60+I).EQ.0.0) GOTO 20
      ISUM=ISUM+1
      IF(ISUM.GE.IORD(6)) GOTO 40
      GOTO 30
 20   ISUM=0
 30   CONTINUE
      RETURN
C
C  CLEAR TRACE AND SET ITYPE=2
C
 40   CALL VCLR(IA,1,NSAMP)
      CALL APWAIT
      ITYPE=2
      RETURN
      END




      SUBROUTINE TMUTE(IDCDP,ITRIN,IORD)
C
      INTEGER*2 IORD(12)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(109),IDELAY),(OUTBUF(111),MUTST),
     +(OUTBUF(113),MUTEND)
C
      MUTE=0
      NTAP=IORD(2)
      IF(ITRIN.LT.IORD(3)) GOTO 200
      IF(IORD(3).LT.0) GOTO 150
      DO 50  I=1,IORD(4)
      J=2*I
      K=2*(I-1)
      IF(ITRIN.LT.IORD(J+3)) GOTO 60
 50   CONTINUE
      GOTO 70
 60   IF(K.EQ.0) GOTO 70
      MUTE=IORD(K+4)+IFIX(FLOAT(IORD(J+4)-IORD(K+4))*
     +FLOAT(ITRIN-IORD(K+3))/FLOAT(IORD(J+3)-IORD(K+3)))
      GOTO 200
 70   MUTE=IORD(J+4)
      GOTO 200
 150  MUTE=IFIX(FLOAT((2*IDCDP)-NTAP)/FSAMP)
 200  LMUT=MAX0(0,ISMPLE(MUTE,FSAMP,IDELAY))
      IF(NTAP.LT.0) GOTO 400
C
```

```
C  MUTE AT START
C
      CALL TAPER(LMUT,1,LMUT,1,NTAP)
      CALL VCLR(IA,1,LMUT)
      CALL APWAIT
      MUTST=IDELAY
      MUTEND=MAX0(MUTE,MUTST)
      RETURN
C
C  MUTE AT END
C
 400  NTAP=IABS(NTAP)
      IF(LMUT.GT.NSAMP) RETURN
      CALL TAPER(LMUT-1,-1,LMUT-1,-1,NTAP)
      CALL VCLR(LMUT,1,NSAMP-LMUT)
      CALL APWAIT
      RETURN
      END



      SUBROUTINE TRAMP(IORD)
C
      REAL*4 CONST(3),FACT
      INTEGER*2 IORD(12),IFACT(2)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/    IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (IFACT(1),FACT),(OUTBUF(109),IDELAY)
C
C  FORM TIME RAMP
C
      DO 3 I=1,2
 3    IFACT(I)=IORD(3+I)
      CONST(1)=0.001*FLOAT(IDELAY)
      CONST(2)=FACT
      CONST(3)=1.0/(1000.0*FSAMP)
      CALL APPUT(CONST,ID,3,2)
      CALL APWD
      CALL VRAMP(ID,ID+2,IC,1,NSAMP)
C
C  CHECK FOR IORD(2), 0=EXP(F*T), 1=T*EXP(F*T), 2=T*T*EXP(F*T), 3=T*T
C
      IF(IORD(2).LE.2) GO TO 5
      CALL VSQ(IC,1,IB,1,NSAMP)
      GO TO 10
C
C  FORM EXP(FACTOR*TIME)
C
 5    CALL VSMUL(IC,1,ID+1,IB,1,NSAMP)
      CALL VEXP(IB,1,IB,1,NSAMP)
      IF(IORD(2).EQ.0) GO TO 10
      CALL VMUL(IC,1,IB,1,IB,1,NSAMP)
      IF(IORD(2).EQ.2) CALL VMUL(IC,1,IB,1,IB,1,NSAMP)
C
C  APPLY OR REMOVE ACCORDING TO IAPPLY FLAG
C
 10   IF(IORD(3).EQ.0) GO TO 20
      CALL VMUL(IA,1,IB,1,IA,1,NSAMP)
      GO TO 30
C
C  CATCH ALL ZEROS IN TIME RAMP BEFORE DIVIDE
C
 20   CALL LVNOT(IB,1,IC,1,NSAMP)
      CALL VADD(IC,1,IB,1,IB,1,NSAMP)
      CALL VDIV(IB,1,IA,1,IA,1,NSAMP)
 30   CALL APWAIT
      RETURN
      END
```

```
      SUBROUTINE TVRAMP(IORD)
C
      REAL*4 CONST(3),T0LYR(10),VLYR(10)
      INTEGER*2 IORD(12),IT0LYR(10),IVLYR(10)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/DATBUF/OUTBUF
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(199),ILYR),(OUTBUF(201),IT0LYR(1)),
     +(OUTBUF(221),IVLYR(1)),(OUTBUF(109),IDELAY)
C
C  GET VELOCITY STRUCTURE
C
      NLYR=ILYR
      IF(NLYR.EQ.0) STOP'NO VELOCITY DATA'
      DO 50 I=1,NLYR
      T0LYR(I)=FLOAT(IT0LYR(I))
 50   VLYR(I)=0.001*FLOAT(IVLYR(I))
C
C  FORM TIME RAMP
C
      CONST(1)=0.001*FLOAT(IDELAY)
      CONST(3)=1.0/(1000.0*FSAMP)
      CALL APPUT(CONST,ID,3,2)
      CALL APWD
      CALL VRAMP(ID,ID+2,IC,1,NSAMP)
C
C  FORM VELOCITY FUNCTION
C VELOCITIES ARE CONSTANT UP TO FIRST REFLECTOR AND BEYOND
C LAST AND ARE LINEARLY INTERPOLATED IN BETWEEN.
C
      N1=IB
      V=VLYR(1)
      CALL APWR
      CALL APPUT(V,N1,1,2)
      CALL APWD
      N=IFIX(FSAMP*(T0LYR(1)-IDELAY))
      IF(N.LE.0) GOTO 5
      IF(N.GT.NSAMP-1) GOTO 21
      CALL VFILL(N1,N1,1,N+1)
 5    IF(NLYR.EQ.1) GOTO 20
      DO 10 J=2,NLYR
      N1=N1+N
      IF(N1.GE.IB-1+NSAMP) GOTO 25
      N=IFIX(FSAMP*(T0LYR(J)-IDELAY))-N1+4096
      IF(N1+N.LE.IB) GOTO 10
      DELV=(VLYR(J)-VLYR(J-1))/N
      IF(N1.GT.IB) GOTO 15
      V=DELV*(IB-N1)+VLYR(J-1)
      CALL APWR
      CALL APPUT(V,IB,1,2)
      CALL APPUT(DELV,IB+1,1,2)
      CALL APWD
      CALL VRAMP(IB,IB+1,IB,1,N1+N-IB-1)
      GOTO 10
 15   CALL APWR
      CALL APPUT(DELV,N1+1,1,2)
      CALL APWD
      IF(N1+N.GE.IB+NSAMP) GOTO 24
      CALL VRAMP(N1,N1+1,N1,1,N+1)
 10   CONTINUE
 20   N1=N1+MAX0(0,N)
 21   N=NSAMP+IB-N1
      CALL VFILL(N1,N1,1,N)
      GOTO 25
 24   CALL VRAMP(N1,N1+1,N1,1,IB+NSAMP-N1)
C
C  FORM TV**2 VECTOR
C
 25   CALL VMUL(IC,1,IB,1,IB,1,NSAMP)
```

```
      CALL VSQ(IB,1,IB,1,NSAMP)
C
C  APPLY OR REMOVE ACCORDING TO IAPPLY FLAG
C
      IF(IORD(2).EQ.0) GO TO 30
      CALL VMUL(IA,1,IB,1,IA,1,NSAMP)
      GO TO 40
C
C  CATCH ALL ZEROS IN TIME RAMP BEFORE DIVIDE
C
 30   CALL LVNOT(IB,1,IC,1,NSAMP)
      CALL VADD(IC,1,IB,1,IB,1,NSAMP)
      CALL VDIV(IB,1,IA,1,IA,1,NSAMP)
 40   CALL APWAIT
      RETURN
      END




      SUBROUTINE AGC(IDCDP,IORD)
C
      INTEGER*2 IORD(12)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(109),IDELAY)
C
C  SET UP INITIAL CONDITIONS
C
      ISTSMP=ISMPLE(IDELAY,FSAMP,0)
      IHWIN=IABS(IORD(2))
      IF(IORD(2).LT.0) IORD(3)=MAX0(0,2*IDCDP+IHWIN-ISTSMP)
      CALL VSQ(IA,1,IB,1,NSAMP)
      CALL VRNAVG(IB,IC,NSAMP,IHWIN)
      CALL VFILL(IC+IORD(3),IC,1,IORD(3))
      CALL VFILL(ID-IHWIN-1,ID-IHWIN,1,IHWIN)
      CALL VSQRT(IC,1,IC,1,NSAMP)
C
C  REMOVE ANY ZEROES FROM NORM VECTOR BY SETTING TO 1.0
C
      CALL LVNOT(IC,1,IB,1,NSAMP)
      CALL VADD(IB,1,IC,1,IC,1,NSAMP)
C
C  APPLY TO DATA
C
      CALL VDIV(IC,1,IA,1,IA,1,NSAMP)
C
C  RETURN
C
      CALL APWAIT
      RETURN
      END
```

```
      SUBROUTINE BANDPS(IORD)
C
      INTEGER*2 IORD(12)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      NSAMP4=2*NSAMP2
      NCLR=NSAMP4+2-NSAMP
      CALL VCLR(IB,1,NCLR)
      CALL TAPER(IA,1,IA,1,10)
      CALL TAPER(IB-1,-1,IB-1,-1,10)
      CALL RFFT(IA,NSAMP4,1)
      CALL RFFTSC(IA,NSAMP4,3,1)
      CALL VCLR(IA,1,IORD(2))
      CALL CTAP(IORD(2),IORD(2),1,IORD(7))
      CALL CTAP(IORD(3),IORD(3),0,IORD(6))
      CALL VCLR(IORD(4),1,IORD(5))
      CALL RFFTSC(IA,NSAMP4,-3,0)
      CALL RFFT(IA,NSAMP4,-1)
      CALL APWAIT
      RETURN
      END
      SUBROUTINE BANDRJ(IORD)
C
      INTEGER*2 IORD(12)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      NSAMP4=2*NSAMP2
      NCLR=NSAMP4+2-NSAMP
      CALL VCLR(IB,1,NCLR)
      CALL TAPER(IA,1,IA,1,10)
      CALL TAPER(IB-1,-1,IB-1,-1,10)
      CALL RFFT(IA,NSAMP4,1)
      CALL RFFTSC(IA,NSAMP4,3,1)
      CALL CTAP(IORD(2),IORD(2),0,IORD(4))
      CALL CTAP(IORD(3),IORD(3),1,IORD(4))
      CALL RFFTSC(IA,NSAMP4,-3,0)
      CALL RFFT(IA,NSAMP4,-1)
      CALL APWAIT
      RETURN
      END




      SUBROUTINE DESIGN(FILTER,NBEG,LAG,ICDPNO,ITRIN,IDCDP,IORD)
C
      VIRTUAL FILTER(4096),NBEG(3,96)
      INTEGER*2 IORD(12)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(111),MUTST),(OUTBUF(113),MUTEND),
     +(OUTBUF(109),IDELAY)
C
      LAG=IORD(5)
      IF(LAG.LT.0) LAG=(2*IDCDP)+IORD(5)
      IORD(8)=IORD(4)+LAG
      IF(IORD(8).LT.IORD(6)) GOTO 525
      TYPE 5200,IORD(2),ICDPNO,ITRIN
      WRITE(2,5200) IORD(2),ICDPNO,ITRIN
 5200 FORMAT(' DECON FILTER',I2,' NOT APPLIED TO CDP',I6,' TRACE',I3,/,
     +' BECAUSE TOTAL FILTER LENGTH EXCEEDS AUTO-CORR DESIGN WINDOW.'/)
      IORD(9)=-1
      RETURN
```

```
 525  IF(IORD(3).LE.1)
     +IBEG=MAX0(0,ISMPLE(NBEG(IORD(2),1),FSAMP,IDELAY))
      IF(IORD(3).EQ.2)
     +IBEG=MAX0(0,ISMPLE((MUTEND+NBEG(IORD(2),1)),FSAMP,IDELAY))
      IF(IORD(3).EQ.3)
     +IBEG=MAX0(0,ISMPLE(NBEG(IORD(2),ITRIN),FSAMP,IDELAY))
      IF(IORD(10).NE.0) RETURN
C
      WHITE=1.0+(0.01*FLOAT(IORD(7)))
      NLG=ID+LAG
C
      CALL VCLR(IB,1,IE-IB)
C
      CALL VMOV(IBEG,1,IB,1,IORD(6))
      CALL CONV(IB,1,IB,1,ID,1,IORD(8),IORD(6))
      CALL VCLR(IB,1,ID-IB)
      CALL APWAIT
      CALL APPUT(WHITE,IB,1,2)
      CALL APWD
      CALL VSMUL(ID,1,IB,ID,1,1)
C
      CALL WIENER(IORD(4),ID,NLG,IC,IB,1)
      CALL APWR
      CALL APCHK(IORD(9))
      CALL APWD
      IF(IORD(9).EQ.0) GOTO 10
C
C  LEVINSON FAILURE
C
      WRITE(2,2500) IORD(9),ICDPNO,ITRIN
      TYPE 2500,IORD(9),ICDPNO,ITRIN
 2500 FORMAT(' LEVINSON FAILURE AT ITERATION:',I5,
     +' CDP NO.',I5,' TRACE NO.'I3/)
      RETURN
C
  10  CALL VCLR(ID,1,IORD(8))
      CALL VTSADD(ID,1,2049,ID,1,1)
      CALL VSUB(IC,1,NLG,1,NLG,1,IORD(4))
      CALL APWAIT
      IF(IORD(3).NE.0) RETURN
C
      CALL APGTXM(FILTER(1),ID,IORD(8),2)
      CALL APWAIT
C
      IORD(10)=1
      RETURN
      END



      SUBROUTINE APPLY(FILTER,IORD)
C
C THIS ROUTINE APPLIES A GIVEN FILTER
C
      VIRTUAL FILTER(4096)
      INTEGER*2 IORD(12)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      NCLR=ID-NSAMP
      CALL VCLR(IB,1,NCLR)
      IF(IORD(3).NE.0) GOTO 100
C
C  GET FILTER COEFFICIENTS
C
      CALL APWAIT
      CALL APPTXM(FILTER(1),ID,IORD(8),2)
C
C APPLY FILTER
```

```
C
 100  CALL APWAIT
      CALL VMOV(IA,1,IB-1+IORD(8),1,NSAMP)
      CALL VCLR(IA,1,NSAMP)
      CALL CONV(IB,1,ID-1+IORD(8),-1,IA,1,NSAMP,IORD(8))
      CALL APWAIT
      RETURN
      END



      SUBROUTINE AUTOUT(AUTO,NAUBEG,NTRACI,ITRIN,IRTBLK,ITA,IORD,
     +ISTOP)
C
      VIRTUAL AUTO(4096),NAUBEG(2,96)
      INTEGER*2 IORD(12),IRTBLK(2),ITA(2)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(29),ITYPE),(OUTBUF(31),IFOLD),
     +(OUTBUF(35),ITRCIN),(OUTBUF(109),IDELAY),
     +(OUTBUF(111),MUTST),(OUTBUF(113),MUTEND),(OUTBUF(115),NSMP)
C
C  IF FIRST CALL WRITE OUT HEADER BLOCK
C
      IF(IRTBLK(IORD(2)).GT.1) GOTO 50
      LSMP=NSMP
      LTYPE=ITYPE
      ITRCIN=NTRACI
      IF(IFOLD.GT.0) ITRCIN=1
      NSMP=IORD(4)
      ITYPE=9
      IF(IWRITW(120,OUTBUF,0,12+IORD(2)).LT.0) STOP'AUTO H/BLK ERR'
      NSMP=LSMP
      ITYPE=LTYPE
      ITRCIN=0
C
 50   IF(IORD(3).LE.1)
     +IAUBEG=MAX0(0,ISMPLE(NAUBEG(IORD(2),1),FSAMP,IDELAY))
      IF(IORD(3).EQ.2)
     +IAUBEG=MAX0(0,ISMPLE((MUTEND+NAUBEG(IORD(2),1)),FSAMP,IDELAY))
      IF(IORD(3).EQ.3)
     +IAUBEG=MAX0(0,ISMPLE(NAUBEG(IORD(2),ITRIN),FSAMP,IDELAY))
      CALL VCLR(IB,1,IE-IB)
      CALL VMOV(IAUBEG,1,IB,1,IORD(4))
      CALL CONV(IB,1,IB,1,ID,1,IORD(4),IORD(4))
      CALL VCLR(IB,1,ID-IB)
      CALL VMOV(ID,1,IB,1,1)
      CALL VSDIV(ID,1,IB,IC,1,IORD(4))
      CALL APWAIT
      CALL APGTXM(AUTO(1),IC,IORD(4),2)
      CALL APWAIT
C
C  WRITE TO DISK FILE
C
      IWT=IWDRXM(AUTO(1),12+IORD(2),2*IORD(4),IRTBLK(IORD(2)),0)
      IF(IWT.EQ.2*IORD(4)) GOTO 100
      ISTOP=32
      RETURN
 100  IRTBLK(IORD(2))=IRTBLK(IORD(2))+IORD(5)
      ITA(IORD(2))=ITA(IORD(2))+IORD(5)
      RETURN
      END
```

```
      SUBROUTINE HILBRT
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      IH=IA+NSAMP2
      L=NSAMP2/2
      NCLR=2*NSAMP2-NSAMP
      CALL VCLR(NSAMP,1,NCLR)
C      CALL HLBRT(IA,IH,NSAMP2)
      CALL TAPER(IA,1,IA,1,10)
      CALL TAPER(IB-1,-1,IB-1,-1,10)
      CALL RFFT(IA,NSAMP2,1)
      CALL RFFTSC(IA,NSAMP2,IA,1)
      CALL VCLR(IA,1,2)
      CALL VMOV(IA,2,IH+1,2,L)
      CALL VMOV(IA+1,2,IH,2,L)
      CALL VNEG(IH+1,2,IH+1,2,L)
      CALL RFFT(IA,NSAMP2,-1)
      CALL RFFT(IH,NSAMP2,-1)
      CALL VSQ(IA,1,IA,1,NSAMP)
      CALL VSQ(IH,1,IH,1,NSAMP)
      CALL VADD(IA,1,IH,1,IA,1,NSAMP)
      CALL VSQRT(IA,1,IA,1,NSAMP)
      CALL APWAIT
      RETURN
      END


      SUBROUTINE STKSET(V,T0LYR,VLYR,ICDPNO,IORD)
C
      VIRTUAL V(2,11,100),T0LYR(10),VLYR(10)
      INTEGER*2 IORD(12)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
C
      EQUIVALENCE (OUTBUF(109),IDELAY)
C
C  SET UP CONSTANTS
C
      IORD(10)=0
      IF(IORD(2).EQ.2) RETURN
      TREND=FLOAT(IDELAY)+FLOAT(NSAMP)/FSAMP
 10   N1VAN=IFIX(V(1,1,IORD(9)))
      IORD(7)=IFIX(V(2,1,IORD(9)))
      DO 20 I=1,IORD(7)
      T0LYR(I)=V(1,I+1,IORD(9))
 20   VLYR(I)=V(2,I+1,IORD(9))
      IF(IORD(9).EQ.IORD(6)) GOTO 50
      N2VAN=IFIX(V(1,1,IORD(9)+1))
      IF(ICDPNO.LE.N1VAN) GOTO 50
      IF(ICDPNO.LT.N2VAN) GOTO 30
      IORD(9)=IORD(9)+1
      GOTO 10
 30   IF(IORD(5).EQ.0) GOTO 50
      RINT=FLOAT(N2VAN-N1VAN)
      IF(RINT.LE.1.0) GOTO 50
      DO 40 I=1,IORD(7)
      T0LYR(I)=T0LYR(I)+(V(1,I+1,IORD(9)+1)-V(1,I+1,IORD(9)))*
     *FLOAT(ICDPNO-N1VAN)/RINT
 40   VLYR(I)=VLYR(I)+(V(2,I+1,IORD(9)+1)-V(2,I+1,IORD(9)))*
     *FLOAT(ICDPNO-N1VAN)/RINT
C
C  SET UP RMS VELOCITY / TIME TABLE
C
 50   IF(T0LYR(IORD(7)).GE.TREND) GOTO 80
      IF(IORD(7).LT.10) GOTO 60
      VLYR(10)=VLYR(10)+((VLYR(10)-VLYR(9))*
     $(TREND-T0LYR(10))/(T0LYR(10)-T0LYR(9)))
```

```
      GOTO 70
 60   IORD(7)=IORD(7)+1
      VLYR(IORD(7))=VLYR(IORD(7)-1)
 70   T0LYR(IORD(7))=TREND
 80   CONTINUE
      RETURN
      END


      SUBROUTINE STACK(T0LYR,VLYR,X,XM,NTRACI,ITRIN,IORD)
C
      VIRTUAL T0LYR(10),VLYR(10),X(96),XM(22)
      INTEGER*4 JOFF
      INTEGER*2 IORD(12)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/DATBUF/OUTBUF
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(37),JOFF),(OUTBUF(111),MUTST),
     +(OUTBUF(113),MUTEND),(OUTBUF(109),IDELAY)
C
      IF(IORD(2).EQ.2) GOTO 310
      MSAMP2=IORD(3)*NSAMP2
      IX=IB+MSAMP2+2
      IM=IX+1
      IV=IM+1
      IT=IV+10
C
C  GET CHANNEL OFFSET
C
      IDEL=ISMPLE(IDELAY,FSAMP,0)
      MDEL=IORD(3)*IDEL
      MSAMP=IORD(3)*NSAMP
      CALL JJCVT(JOFF)
      CALL IJCVT(JOFF,IOFF)
      CALL JJCVT(JOFF)
      XM(1)=X(IORD(8)*NTRACI+ITRIN)
      IF(XM(1).LT.0.0) XM(1)=FLOAT(IOFF)
      XM(1)=ABS(XM(1))
      IMUTE=ISMPLE(MUTEND,FSAMP,MUTST)*IORD(3)
      XM(2)=FLOAT(IORD(3))
C
C  CONVERT TIME AND VELOCITY TO SAMPLE TIME
C
      DO 205 I=1,IORD(7)
      XM(12+I)=FLOAT(IFIX(T0LYR(I)*FSAMP))
 205  XM(2+I)=VLYR(I)/FSAMP
C
C  PUT CONSTANTS INTO AP
C
      CALL APWAIT
      CALL APPTXM(XM(1),IX,22,2)
      CALL APWAIT
C
C  NEW INTERPOLATION USING FREQUENCY DOMAIN ONLY
C
      CALL VCLR(IB,1,MSAMP2+2)
      CALL TAPER(IA,1,IA,1,10)
      CALL TAPER(IB-1,-1,IB-1,-1,10)
      CALL VMOV(IA,1,IB,1,NSAMP)
      CALL RFFT(IB,NSAMP2,1)
      CALL RFFTSC(IB,NSAMP2,3,1)
      CALL RFFTSC(IB,MSAMP2,-3,0)
      CALL RFFT(IB,MSAMP2,-1)
      CALL APWR
C
C  DO NMO
C
      IF(IORD(2).LT.-1) GOTO 330
```

```
      CALL NMO(IX,IM,IV,IT,MDEL,IB,MSAMP,IDEL,IA,NSAMP,IG,IMUTE)
      IF(IORD(2).LT.0) GOTO 400
C
C  ADD TO STACK VECTOR
C
 310  IORD(10)=IORD(10)+1
      CALL VADD(IA,1,IE,1,IE,1,NSAMP)
      GOTO 400
C
C  DO REVERSE MOVE-OUT
C
 330  CALL OMN(IX,IM,IV,IT,MDEL,IB,MSAMP,IDEL,IA,NSAMP)
 400  CALL APWAIT
      RETURN
      END


      SUBROUTINE TMIX(SEISA,SEISB,ICDPNO,IORD)
C
      VIRTUAL SEISA(4156),SEISB(4156)
      REAL*4 R(3),TRHDBF(60)
      INTEGER*4 JCDPNO
      INTEGER*2 IORD(12),IR(6)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(1),TRHDBF(1)),(OUTBUF(21),JCDPNO)
      EQUIVALENCE (IR(1),R(1))
C
      IORD(10)=IORD(10)+1
      DO 5 I=1,6
 5    IR(I)=IORD(1+I)
      CALL APPUT(R,ID,3,2)
      CALL APPTXM(SEISA(61),IB,NSAMP,2)
      CALL APPTXM(SEISB(61),IC,NSAMP,2)
      CALL APGTXM(SEISA(61),IA,NSAMP,2)
      CALL APGTXM(SEISB(61),IB,NSAMP,2)
      CALL APWAIT
C
C  IF FIRST TRACE EXIT
C
      IF(IORD(10).EQ.1) GOTO 10
C
C  CLEAR TOP BUFFER IF SECOND TRACE
C
      IF(IORD(10).EQ.2) CALL VCLR(IC,1,NSAMP)
      CALL APWR
      CALL VSMUL(IA,1,ID+2,IA,1,NSAMP)
      CALL VSMUL(IB,1,ID+1,IB,1,NSAMP)
      CALL VSMUL(IC,1,ID,IC,1,NSAMP)
      CALL VADD(IA,1,IB,1,IA,1,NSAMP)
      CALL VADD(IA,1,IC,1,IA,1,NSAMP)
      CALL APWAIT
C
C  GET HEADER BLOCKS SORTED OUT
C
 10   DO 20 I=1,60
      SEISB(I)=SEISA(I)
      SEISA(I)=TRHDBF(I)
 20   TRHDBF(I)=SEISB(I)
C
C  RETURN
C
      CALL JJCVT(JCDPNO)
      CALL IJCVT(JCDPNO,ICDPNO)
      CALL JJCVT(JCDPNO)
      RETURN
      END
```

```
      SUBROUTINE VELFIL(SEIS,T0LYR,VLYR,X,XM,TRHD,RESULT,ICDPNO,
     +IORD,ISTOP)
C
      VIRTUAL SEIS(4156),T0LYR(10),VLYR(10),X(96),XM(22),RESULT(32),
     +TRHD(96,60)
      INTEGER*2 IORD(12)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/INPUT/ NFILES,NCHAN,NSMPIN,ISTART,ILEN,IFIRST,ILAST,
     +FTRCNO,IWORDR,IWORDW,IBLK,IVFRT,IDPRT,NTRACI
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      LOOP=((IORD(11)-1)/2)
      LOOP1=LOOP+1
      ITRI=0
      ITRO=0
      IA=IORD(11)*NSAMP
      IB=IA+NSAMP+IORD(11)
      IC=IB+IORD(3)*NSAMP2+2
      ID=IC+22
      IE=ID+(IORD(11)*IORD(11))-1
      IG=-1
      IADN=IA+IFIX(IORD(11)/2)
      IADR=LOOP*NSAMP
      CALL VCLR(0,1,IA)
      CALL APWR
      CALL OPERVF(RESULT,IORD(11),ID,IORD(12))
      DO 820 I=1,IORD(7)
 820  VLYR(I)=0.01*FLOAT(IORD(12))*VLYR(I)
      DO 840 I=1,LOOP1
      ITRI=ITRI+1
      CALL DRNMO(SEIS,TRHD,T0LYR,VLYR,X,XM,ITRI,IADR,IORD,ISTOP)
      IF(ISTOP.NE.0) RETURN
 840  IADR=IADR+NSAMP
      IADR=IA-NSAMP
C
C  DO CONVOLUTION
C
 850  CALL IWAIT(IVFRT)
      CALL VCLR(IA,1,NSAMP)
      CALL CSQ2D(0,1,IE,-1,IA,1,NSAMP,IORD(11))
      CALL APWR
      ITRO=ITRO+1
      CALL DROMN(SEIS,TRHD,T0LYR,VLYR,X,XM,ITRO,IADN,IORD,ISTOP)
      IF(ISTOP.NE.0) RETURN
      ITRI=ITRI+1
      IF(ITRI.GT.NTRACI) GOTO 860
      CALL DRNMO(SEIS,TRHD,T0LYR,VLYR,X,XM,ITRI,IADR,IORD,ISTOP)
      IF(ISTOP.NE.0) RETURN
      GOTO 850
C
C  FLUSH OUT LAST TRACES
C
 860  DO 870 I=1,LOOP
      CALL VCLR(IADR,1,NSAMP)
      CALL VCLR(IA,1,NSAMP)
      CALL CSQ2D(0,1,IE,-1,IA,1,NSAMP,IORD(11))
      CALL APWR
      ITRO=ITRO+1
      CALL DROMN(SEIS,TRHD,T0LYR,VLYR,X,XM,ITRO,IADN,IORD,ISTOP)
      IF(ISTOP.NE.0) RETURN
 870  CONTINUE
C
C  RETURN
C
      IA=0
      IB=IA+NSAMP
      IC=IB+NSAMP
      ID=IC+NSAMP
      IG=24576-NSAMP
```

```
      IE=IG-NSAMP
      RETURN
      END

      SUBROUTINE DRNMO(SEIS,TRHD,T0LYR,VLYR,X,XM,ITRI,IADR,IORD,ISTOP)
C
      VIRTUAL SEIS(4156),TRHD(96,60),T0LYR(10),VLYR(10),X(96),XM(22)
      REAL*4 TRHDBF(60)
      INTEGER*2 IORD(12)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/INPUT/ NFILES,NCHAN,NSMPIN,ISTART,ILEN,IFIRST,ILAST,
     +FTRCNO,IWORDR,IWORDW,IBLK,IVFRT,IDPRT,NTRACI
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(1),TRHDBF(1))
C
      IRD=IRDRXM(SEIS(1),IVFRT,IWORDW,(ITRI-1)*IBLK,0)
      IF(IRD.EQ.IWORDW) GOTO 822
      ISTOP=41
      RETURN
 822  DO 824 J=1,60
      TRHDBF(J)=SEIS(J)
 824  TRHD(ITRI,J)=SEIS(J)
      CALL APWAIT
      CALL APPTXM(SEIS(61),IA,NSAMP,2)
      CALL APWAIT
      IORD(2)=-1
      CALL STACK(T0LYR,VLYR,X,XM,NTRACI,ITRI,IORD(1))
      CALL VMOV(IA,1,IADR,1,NSAMP)
      CALL APWAIT
C
C  RETURN
C
      RETURN
      END

      SUBROUTINE DROMN(SEIS,TRHD,T0LYR,VLYR,X,XM,ITRO,IADN,IORD,ISTOP)
C
      VIRTUAL SEIS(4156),TRHD(96,60),T0LYR(10),VLYR(10),X(96),XM(22)
      REAL*4 TRHDBF(60)
      INTEGER*2 IORD(12)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/INPUT/ NFILES,NCHAN,NSMPIN,ISTART,ILEN,IFIRST,ILAST,
     +FTRCNO,IWORDR,IWORDW,IBLK,IVFRT,IDPRT,NTRACI
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(1),TRHDBF(1))
C
      CALL APWAIT
      CALL VMOV(IADN,1,IA,1,NSAMP)
      DO 852 J=1,60
      TRHDBF(J)=TRHD(ITRO,J)
 852  SEIS(J)=TRHD(ITRO,J)
      CALL APWR
      IORD(2)=-2
      CALL STACK(T0LYR,VLYR,X,XM,NTRACI,ITRO,IORD(1))
      CALL APWAIT
      CALL APGTXM(SEIS(61),IA,NSAMP,2)
      CALL APWAIT
      IWT=IWDRXM(SEIS(1),IVFRT,IWORDW,(ITRO-1)*IBLK,0)
      IF(IWT.EQ.IWORDW) RETURN
      ISTOP=42
      RETURN
      END
```

```
      SUBROUTINE DIPFIL(SEIS,DIP,ICDPNO,IORD,ISTOP)
C
      VIRTUAL SEIS(4156),DIP(1024)
      REAL*4 TRHDBF(60)
      INTEGER*4 JCDPNO
      INTEGER*2 IORD(12)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/INPUT/ NFILES,NCHAN,NSMPIN,ISTART,ILEN,IFIRST,ILAST,
     +FTRCNO,IWORDR,IWORDW,IBLK,IVFRT,IDPRT,NTRACI
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
C
      EQUIVALENCE (JCDPNO,OUTBUF(21)),(TRHDBF(1),OUTBUF(1))
C
C  SET UP AP ADDRESSES
C
      IFIL=IORD(10)
      I0=0
      IA=IORD(2)*NSAMP
      ID=IA+NSAMP+IORD(2)
      IG=ID+IORD(2)*IORD(2)-1
      IAN=IA+IORD(9)
      IADR=I0
C
C  LOAD OPERATOR
C
      CALL APPTXM(DIP(1),ID,IORD(2)*IORD(2),2)
      CALL APWAIT
C
C  CLEAR AP AND FILL WITH TRACES FROM DISK
C
      CALL VCLR(I0,1,IA)
      CALL APWR
      DO 100 I=1,IORD(2)
      NBLK=(MOD((IFIL+I-IORD(9)-1),IORD(2))*IBLK)
      IRD=IRDRXM(SEIS(1),IDPRT,IWORDW,NBLK,0)
      IF(IRD.EQ.IWORDW) GOTO 120
      ISTOP=51
      RETURN
 120  IF(I.NE.IORD(9)+1) GOTO 140
      DO 150 J=1,60
 150  TRHDBF(J)=SEIS(J)
      CALL JJCVT(JCDPNO)
      CALL IJCVT(JCDPNO,ICDPNO)
      CALL JJCVT(JCDPNO)
 140  CALL APWAIT
      CALL APPTXM(SEIS(61),IADR,NSAMP,2)
      CALL APWAIT
      IADR=IADR+NSAMP
 100  CONTINUE
C
C  DO CONVOLUTION
C
      CALL APWAIT
      CALL VCLR(IA,1,NSAMP+IORD(2))
      CALL CSQ2D(I0,1,IG,-1,IA,1,NSAMP,IORD(2))
      CALL VMOV(IAN,1,I0,1,NSAMP)
      CALL APWAIT
C
C  RETURN
C
      RETURN
      END


      SUBROUTINE OPERDP(DIP,IORD)
C
C  R W HOBBS AUG 1984
C
      VIRTUAL DIP(1024)
      INTEGER*2 IORD(12),IVAP(2),IFNQ(2),ISNQ(2)
```

```
C
      EQUIVALENCE (IVAP(1),VAP),(IFNQ(1),FNQ),(ISNQ(1),SNQ)
C
      IOP=IORD(2)
      DO 40 I=1,2
      IVAP(I)=IORD(2+I)
      IFNQ(I)=IORD(4+I)
 40   ISNQ(I)=IORD(6+I)
      IF(VAP.LE.0.0) VAP=2.0*FNQ/SNQ
C
      SLOPE=VAP*SNQ/FNQ
      IA=0
      IB=4096
      IC=8192
      ID=12288
      IE=16384
      NOP=64
      NOPH=NOP/2
      NOPH1=NOPH+1
      NOP2=NOPH1*2
      CONST=1.0
      CALL APPUT(CONST,IE,1,2)
      CALL APWD
C
      CALL VFILL(IE,IC,1,NOPH1)
      IKS=MAX0(1,IFIX(FLOAT(NOPH)/SLOPE))
      IF(IKS.GT.NOPH) GOTO 80
      CALL TAPER((IC+2*IKS),-1,(IC+2*IKS),-1,IKS)
      IF(2*IKS.LT.NOPH) CALL VCLR((IC+IKS+1),1,(NOPH-2*IKS))
C
 80   CALL VCLR(IA,1,(NOPH1*NOP2))
      DO 100 I=1,NOPH1
 100  CALL VFILL(IC-1+I,IA+((I-1)*NOP2),2,NOPH1)
      CALL APWR
C
      CONST=0.5
      CALL APPUT(CONST,IE,1,2)
      CALL APWD
      CALL VSMUL(IA,1,IE,IA,1,1)
      CALL VCLR(NOP2,NOP2,NOPH)
      CALL APWR
C
      DO 200 I=1,NOPH1
      IKS=I-1
      IADR=IA+(IKS*NOP2)
      IFS=2*IFIX(SLOPE*FLOAT(IKS))
      NCLR=2*(IFIX(IFS/4))
      NTAP=(IFS-NCLR)/2
      IF(NCLR.GT.NOP2) NCLR=NOP2
      IF((NTAP*2+NCLR).GT.NOP2) NTAP=(NOP2-NCLR)/2
      N1=IADR+NCLR
      IF(NCLR.GT.0) CALL VCLR(IADR,1,NCLR)
      IF(NTAP.GT.0) CALL CTAP(N1,N1,1,NTAP)
 200  CONTINUE
C
      DO 400 I=1,NOPH1
      CALL RFFTSC(IA+((I-1)*NOP2),NOP,-3,1)
 400  CALL RFFT(IA+((I-1)*NOP2),NOP,-1)
      CALL APWR
C
      CALL VCLR(IC,1,(NOPH1*NOPH1*2))
      DO 500 I=1,NOPH1
 500  CALL VMOV(IA+((I-1)*NOP2),1,IC+((I-1)*2),NOP2,NOPH1)
      CALL APWR
C
      DO 600 I=1,NOPH1
      CALL RFFTSC(IC+((I-1)*NOP2),NOP,-3,1)
 600  CALL RFFT(IC+((I-1)*NOP2),NOP,-1)
      CALL APWR
C
      DO 700 I=1,IOP/2+1
      IDES=IE+(IOP*IOP)/2+((I-1)*IOP)
```

```
      CALL VMOV(IC+I-1,NOP2,IDES,1,IOP/2+1)
      CALL VMOV(IDES,1,IDES,-1,IOP/2+1)
      CALL VMOV(IDES-IOP/2,1,IDES-IOP/2-2*((I-1)*IOP),1,IOP)
      CALL APWR
 700  CONTINUE
C
      CONST=4.0
      CALL APPUT(CONST,IC,1,2)
      CALL APWD
      CALL VSMUL(IE,1,IC,IE,1,IOP*IOP)
      CALL VCLR(IA,1,IE)
      CALL APWAIT
C
C  GET OPERATOR INTO DIP ARRAY
C
      CALL APGTXM(DIP(1),IE,IOP*IOP,2)
      CALL APWD
D      TYPE *,(DIP(I),I=1,IOP*IOP)
C
      RETURN
      END




      SUBROUTINE TVBAND(IDCDP,IORD)
C
      REAL*4 A(5),B(5)
      INTEGER*2 IORD(12),JB1(2),IOM1(2),JB2(2),IOM2(2)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (JB1(1),B1),(IOM1(1),OM1),(JB2(1),B2),
     +(IOM2(1),OM2),(OUTBUF(109),IDELAY)
C
      NCLR=IE-IB
      DO 100 I=1,2
      JB1(I)=IORD(2+I)
      IOM1(I)=IORD(4+I)
      JB2(I)=IORD(7+I)
 100  IOM2(I)=IORD(9+I)
C
C  SET UP START AND END CONSTANTS IN SAMPLE SPACE
C
      ITM1=IORD(2)
      ITM2=IORD(7)
      ISTSMP=ISMPLE(IDELAY,FSAMP,0)
      IF(ITM1.LT.0) ITM1=2*IDCDP-ITM1
      IF(ITM2.LT.0) ITM2=2*IDCDP-ITM2
      ITM1=MAX0(0,ITM1-ISTSMP)
      ITM2=MAX0(0,ITM2-ISTSMP)
      IF(ITM2.LE.ITM1) ITM2=ITM1+1
C
C  SET UP ARRAYS FOR INPUT TO AP
C
      A(1)=(0.5*B1)/(1.0+B1+OM1)
      A(2)=0.0
      A(3)=-A(1)
      A(4)=2.0*(OM1-1.0)/(1.0+B1+OM1)
      A(5)=1.0-(4.0*A(1))
      B(1)=(0.5*B2)/(1.0+B2+OM2)
      B(4)=2.0*(OM2-1.0)/(1.0+B2+OM2)
      B(5)=1.0-(4.0*B(1))
      B(1)=(B(1)-A(1))/FLOAT(ITM2-ITM1)
      B(2)=0.0
      B(3)=-B(1)
      B(4)=(B(4)-A(4))/FLOAT(ITM2-ITM1)
      B(5)=(B(5)-A(5))/FLOAT(ITM2-ITM1)
```

```
C
      CALL VCLR(IB,1,NCLR)
      CALL APWR
      CALL APPUT(A,ID+12,5,2)
      CALL APPUT(B,ID+17,5,2)
      CALL APWD
      CALL VMOV(IB-1,-1,IB+1,-1,NSAMP)
      CALL VCLR(IA,1,2)
      CALL TVBP(IA+2,IB+6,ID+12,ID+17,ITM1,ITM2,NSAMP,1)
      CALL VNEG(ID+17,1,ID+17,1,5)
      CALL TVBP(IB+2,ID+10,ID+12,ID+17,ITM2,ITM1,NSAMP,-1)
      CALL VADD(IB+6,1,IC+10,1,IA,1,NSAMP)
C
      CALL APWAIT
      RETURN
      END



      SUBROUTINE HSUM(SSUM,IORD)
C
      VIRTUAL SSUM(4096)
      INTEGER*2 IORD(12)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/ OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(29),ITYPE),(OUTBUF(33),ISUM)
C
      IORD(3)=0
      ISUM=0
      IF(ITYPE.EQ.1) IORD(4)=IORD(4)+1
      IORD(5)=IORD(5)+1
      IF(IORD(5).EQ.1) GOTO 100
C
 50   CALL APWAIT
      CALL APPTXM(SSUM(1),IB,NSAMP,2)
      CALL APWAIT
      CALL VADD(IA,1,IB,1,IA,1,NSAMP)
      CALL APWR
      IF(IORD(5).EQ.IORD(2)) GOTO 200
 100  CALL APWAIT
      CALL APGTXM(SSUM(1),IA,NSAMP,2)
      CALL APWAIT
      RETURN
C
 200  IORD(3)=1
      SUM=FLOAT(IORD(4))
      IORD(4)=0
      IORD(5)=0
      CALL APPUT(SUM,IC,1,2)
      CALL APWD
      CALL VSDIV(IA,1,IC,IA,1,NSAMP)
      ISUM=IORD(4)
      CALL APWAIT
      RETURN
      END



      SUBROUTINE AUTOST(SEIS,ICDPNO,IORD,ISTOP)
C
      VIRTUAL SEIS(4156)
      REAL*4 TRHDBF(60)
      INTEGER*2 IORD(12),ISTAT(24)
      LOGICAL*1 OUTBUF(240)
```

```
C
      COMMON/INPUT/ NFILES,NCHAN,NSMPIN,ISTART,ILEN,IFIRST,ILAST,
     +FTRCNO,IWORDR,IWORDW,IBLK,IVFRT,IDPRT,NTRACI
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/DATBUF/OUTBUF
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(1),TRHDBF(1))
      EQUIVALENCE (OUTBUF(29),ITYPE),(OUTBUF(109),IDELAY),
     +(OUTBUF(103),ITOTST)
C
C  FORM PILOT TRACE
C
      ITIME=0
 50   CALL VCLR(IA,1,4*NSAMP)
      DO 100 I=IORD(2),IORD(3)
      NBLK=(I-1)*IBLK
      IRD=IRDRXM(SEIS(1),IVFRT,IWORDW,NBLK,0)
      IF(IRD.NE.IWORDW) ISTOP=41
      IF(ISTOP.NE.0) RETURN
      CALL APWAIT
      CALL APPTXM(SEIS(61),IC,NSAMP,2)
      CALL APWAIT
      CALL VADD(IA,1,IC,1,IA,1,NSAMP)
 100  CONTINUE
C
C  CHECK FOR ZERO PILOT TRACE
C
      CALL SVESQ(IA,1,IC,NSAMP)
      CALL APWR
      CALL APGET(TOT,IC,1,2)
      CALL APWD
      IF(TOT.GT.0.0) GOTO 200
      WRITE(2,1000) ICDPNO
 1000 FORMAT(' AUTO-STATICS NOT APPLIED TO CMP ',I5,' BLANK',
     +' PILOT TRACE')
      RETURN
C
C  READ EACH TRACE AND FORM X-CORR FUNCTION
C
 200  DO 400 I=1,NTRACI
      NBLK=(I-1)*IBLK
      IRD=IRDRXM(SEIS(1),IVFRT,IWORDW,NBLK,0)
      IF(IRD.NE.IWORDW) ISTOP=41
      IF(ISTOP.NE.0) RETURN
      DO 250 J=1,60
 250  TRHDBF(J)=SEIS(J)
      IF(ITYPE.NE.1) GOTO 400
      NST=MAX0(0,ISMPLE(IORD(4),FSAMP,IDELAY))
      CALL APWAIT
      CALL APPTXM(SEIS(61),IC,NSAMP,2)
      CALL APWAIT
      IST=NST+IC-IORD(6)
      LEN=2*IORD(6)+1
      CALL CONV(IST,1,NST,1,IB,1,LEN,IORD(5))
C
C  GET MAX X-CORR VALUE
C
      CALL MAXV(IB,1,IB+LEN,LEN)
      CALL APWR
      CALL APCHK(ISHIFT)
      CALL APWD
      ISHIFT=ISHIFT-IB-IORD(6)
      ITOTST=ITOTST+IFIX(FLOAT(ISHIFT)/FSAMP)
      IF(IORD(7).EQ.0.OR.NTRACI.GT.24) GOTO 300
      IF(ITIME.EQ.0) ISTAT(I)=0
      ISTAT(I)=ISTAT(I)+ISHIFT
C
C  APPLY SHIFT TO TRACE
C
 300  CALL APGTXM(SEIS(61),IC+ISHIFT,NSAMP,2)
      CALL APWD
```

```
      DO 350 J=1,60
 350  SEIS(J)=TRHDBF(J)
      IWT=IWDRXM(SEIS(1),IVFRT,IWORDW,NBLK,0)
      IF(IWT.NE.IWORDW) ISTOP=42
      IF(ISTOP.NE.0) RETURN
C
C  END OF TRACE LOOP
C
 400  CONTINUE
      ITIME=ITIME+1
      IF(ITIME.EQ.1) GOTO 50
      IF(IORD(7).EQ.0.OR.NTRACI.GT.24) RETURN
      WRITE(2,555) (ISTAT(I),I=1,NTRACI)
 555  FORMAT(' ',24I4)
      RETURN
      END




      SUBROUTINE TRINPT(SEIS,IDEPTH,IPS,ITVF,ITRI,ITRO,ICDPNO,
     +ICHAN,ITRIN,IDCDP,IDS,IDR,ISTOP)
C
      VIRTUAL IDEPTH(10000),SEIS(4156)
      REAL*4 TRHDBF(60),TMNAM(3)
      INTEGER*4 JORFNO,JORFTR,JCDPNO,JCDPTR
      INTEGER*2 IORD(12,20),REELNU,TPDRR,TPDRW,ITMERR(2),IT0LYR(10),
     +IVLYR(10)
      LOGICAL*1 OUTBUF(240),IANS,LDUM
C
      COMMON/INPUT/ NFILES,NCHAN,NSMPIN,ISTART,ILEN,IFIRST,ILAST,
     +FTRCNO,IWORDR,IWORDW,IBLK,IVFRT,IDPRT,NTRACI
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/CTRL/  ITM,IST,IVF,IDP,IAU,IORD
      COMMON/STATDP/IDFLG,IDPTOT,IDPINT,ISSHOT,IXES,IXSR,IXSS,IXRR
      COMMON/DATBUF/OUTBUF
      COMMON/REELS/ ILINE,REELNU,NTRACO
      COMMON/TAPE/  TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/ADR/   IA,IB,IC,ID,IE,IG
C
      EQUIVALENCE (OUTBUF(9),JORFNO),(OUTBUF(13),JORFTR),
     +(OUTBUF(21),JCDPNO),(OUTBUF(25),JCDPTR),
     +(OUTBUF(29),ITYPE),(OUTBUF(33),IFOLD),(OUTBUF(109),IDELAY),
     +(OUTBUF(111),MUTST),(OUTBUF(113),MUTEND),(OUTBUF(115),NSMP),
     +(OUTBUF(199),ILYR),(OUTBUF(201),IT0LYR(1)),
     +(OUTBUF(221),IVLYR(1))
      EQUIVALENCE (OUTBUF(1),TRHDBF(1))
C
C  READ IN TRACE
C
 61   IF(IPS.EQ.0) GOTO 64
      ITVF=ITVF+1
      IRD=IRDRXM(SEIS(1),IVFRT,IWORDW,(ITVF-1)*IBLK,0)
      IF(IRD.NE.IWORDW) ISTOP=33
      DO 62 I=1,60
 62   TRHDBF(I)=SEIS(I)
      GOTO 68
 64   ITRI=ITRI+1
      CALL TRREAD(SEIS,ICDPNO,IWORDR,ITRI,ITRO,ISTOP)
 68   IF(ISTOP.NE.0) RETURN
      CALL JJCVT(JCDPNO)
      CALL IJCVT(JCDPNO,ICDPNO)
      CALL JJCVT(JCDPNO)
      CALL JJCVT(JCDPTR)
      CALL IJCVT(JCDPTR,ICDPTR)
      CALL JJCVT(JCDPTR)
C
C  CHECK TRACE NUMBER AGAINST LOOP COUNTER
C
      IF(ICDPTR.EQ.ICHAN) GOTO 70
      TYPE 2001,ICDPNO,ICDPTR,ICHAN
```

```
      WRITE(2,2001) ICDPNO,ICDPTR,ICHAN
 2001 FORMAT(/,' • WARNING • TRACE ERROR AT CDP',I6,
     +'  READ',I4,' EXPECTING',I4)
      TYPE 2002
 2002 FORMAT('$CONTINUE (Y/N)? ')
      ACCEPT 2003,IANS
 2003 FORMAT(A1)
      IF(IANS.EQ.'Y') GOTO 75
      ISTOP=31
      RETURN
 75   TYPE 2004
 2004 FORMAT('$(I)NSERT DUMMY TRACE, (Z)ERO OR (S)KIP THIS',
     +' TRACE (I/Z/S) ? ')
      ACCEPT 2003,IANS
      IF(IANS.EQ.'S') GOTO 61
      ITYPE=2
      IF(IANS.EQ.'Z') GOTO 70
      ICDPTR=ICHAN
      CALL BACKSP(1,TPDRR+10)
C
C  CHECK IF TRACE REQUIRED
C
 70   IF(ICDPTR.LT.IFIRST.OR.ICDPTR.GT.ILAST) GOTO 565
      ITRIN=ICDPTR-IFIRST+1
C
C  CHECK IF DEPTHS REQUIRED IF SO CALCULATE
C
      IF(IDFLG.LE.0) GOTO 110
      CALL JJCVT(JORFNO)
      CALL IJCVT(JORFNO,IORFNO)
      CALL JJCVT(JORFNO)
      CALL JJCVT(JORFTR)
      CALL IJCVT(JORFTR,IORFTR)
      CALL JJCVT(JORFTR)
      CALL DEPTH(IDEPTH,IORFNO,IORFTR,IDCDP,IDS,IDR)
 110  IF(IDFLG.GE.0.OR.IST.GT.0) GOTO 120
      IF(ILYR.EQ.0) STOP'NO VEL DATA'
      IDCDP=ISMPLE(IT0LYR(1),FSAMP*0.5,0)
C
C  PUT TRACE INTO AP
C
 120  ITEND=IDELAY+IFIX(FLOAT(NSMP)/FSAMP)
      IBEG=ISTART
      IF(ISTART.LT.0) IBEG=IDELAY
      NSOFF=0
      IAPST=ISMPLE(IDELAY,FSAMP,IBEG)
      IF(IAPST.GE.0) GOTO 130
      NSOFF=-IAPST
      IAPST=0
 130  NSMPUT=IFIX(FLOAT(MIN0((IBEG+ILEN),ITEND)-
     $MAX0(IBEG,IDELAY))*FSAMP)
      IF(ITYPE.NE.1) GOTO 135
      CALL APWAIT
      CALL APPTXM(SEIS(61+NSOFF),IAPST,NSMPUT,2)
      CALL APWAIT
 135  NSMP=NSAMP
      IDELAY=IBEG
C
C  NORMAL RETURN
C
      RETURN
C
C  TRACE OUT OF REQUIRED RANGE
C
 565  ITRIN=-1
      RETURN
      END
```

```
      SUBROUTINE SEISND(ISTOP,ICDPNO)
C
      INTEGER*2 IORD(12,20)
      LOGICAL*1 SEISHD(6),SEISSB(6),TRWRIT(6),TRREAD(6),VELFIL(6),
     +DIPFIL(6)
C
      COMMON/INPUT/ NFILES,NCHAN,NSMPIN,ISTART,ILEN,IFIRST,ILAST,
     +FTRCNO,IWORDR,IWORDW,IBLK,IVFRT,IDPRT,NTRACI
      COMMON/CTRL/  ITM,IST,IVF,IDP,IAU,IORD
C
      CALL SCOPY('SEISHD',SEISHD,6)
      CALL SCOPY('SEISSB',SEISSB,6)
      CALL SCOPY('TRWRIT',TRWRIT,6)
      CALL SCOPY('TRREAD',TRREAD,6)
      CALL SCOPY('VELFIL',VELFIL,6)
      CALL SCOPY('DIPFIL',DIPFIL,6)
C
      IF(ISTOP.EQ.0)  GOTO 500
      IF(ISTOP.GT.50) GOTO 100
      IF(ISTOP.GT.40) GOTO 110
      IF(ISTOP.GT.30) GOTO 120
      IF(ISTOP.GT.20) GOTO 140
      IF(ISTOP.GT.10) GOTO 160
C
      WRITE(2,2000)
      TYPE 2000
 2000 FORMAT(///)
C
      ICDPNO=ICDPNO-1
      TYPE 1000,(SEISHD(I),I=1,6),ISTOP
      WRITE(2,1000)(SEISHD(I),I=1,6),ISTOP
 1000 FORMAT(/' TERMINATION FROM SUBROUTINE ',6A1,' ISTOP=',I2)
      GOTO (1,2,3,4) ISTOP
 160  TYPE 1000,(TRREAD(I),I=1,6),ISTOP
      WRITE(2,1000)(TRREAD(I),I=1,6),ISTOP
      GOTO (11,12,13)(ISTOP-10)
 140  ICDPNO=ICDPNO-1
      TYPE 1000,(TRWRIT(I),I=1,6),ISTOP
      WRITE(2,1000)(TRWRIT(I),I=1,6),ISTOP
      GOTO (21,22)(ISTOP-20)
 120  TYPE 1000,(SEISSB(I),I=1,6),ISTOP
      WRITE(2,1000)(SEISSB(I),I=1,6),ISTOP
      GOTO (31,32,33,34,35)(ISTOP-30)
 110  TYPE 1000,(VELFIL(I),I=1,6),ISTOP
      WRITE(2,1000)(VELFIL(I),I=1,6),ISTOP
      GOTO (41,42)(ISTOP-40)
 100  TYPE 1000,(DIPFIL(I),I=1,6),ISTOP
      WRITE(2,1000)(DIPFIL(I),I=1,6),ISTOP
      GOTO (51)(ISTOP-50)
C
 1    WRITE(2,2001)
      TYPE 2001
 2001 FORMAT(' WRITE DRIVE EXIT')
      GOTO 510
 2    WRITE(2,2002)
      TYPE 2002
 2002 FORMAT(' TAPE HEADER ENCODE ERROR')
      GOTO 510
 3    WRITE(2,2003)
      TYPE 2003
 2003 FORMAT(' ASCII HEADER WRITE ERROR')
      GOTO 510
 4    WRITE(2,2004)
      TYPE 2004
 2004 FORMAT(' BINARY HEADER WRITE ERROR')
      GOTO 510
C
 11   WRITE(2,2011)
      TYPE 2011
 2011 FORMAT(' TRACE READ ERROR')
      GOTO 510
 12   WRITE(2,2012)
```

```
      TYPE 2012
 2012 FORMAT(' READ DRIVE EXIT')
      GOTO 510
 13   WRITE(2,2013)
      TYPE 2013
 2013 FORMAT(' REEL HEADER READ ERROR')
      GOTO 510
 21   WRITE(2,2021)
      TYPE 2021
 2021 FORMAT(' BAD O/P TAPE')
      GOTO 510
 22   WRITE(2,2022)
      TYPE 2022
 2022 FORMAT(' WRITE DRIVE EXIT')
      GOTO 510
C
 31   WRITE(2,2031)
      TYPE 2031
 2031 FORMAT(' BACK WIND BOTH I/P AND O/P TAPES',/,
     +' AND CHECK FOR FULL FOLD OF COVER')
      GOTO 510
 32   WRITE(2,2032)
      TYPE 2032
 2032 FORMAT(' AUTOCORRELATION DISK FILE WRITE ERROR',/,
     +' REWIND BOTH I/P AND O/P TAPES TO START OF CURRENT CDP')
      GOTO 510
 33   WRITE(2,2033)
      TYPE 2033
 2033 FORMAT(' DISK READ ERROR WHILE READING TRACES',
     +' AFTER VELOCITY FILTERING OR AUTOSTATICS.',/,
     +' REWIND BOTH I/P AND O/P TAPES TO START OF CURRENT CDP')
      GOTO 510
 34   WRITE(2,2034)
      TYPE 2034
 2034 FORMAT(' DISK WRITE ERROR WHILE WRITING TRACES',
     +' PRIOR  TO VELOCITY FILTERING.',/,
     +' REWIND BOTH I/P AND O/P TAPES TO START OF CURRENT CDP')
      GOTO 510
 35   WRITE(2,2035)
      TYPE 2035
 2035 FORMAT(' DISK WRITE ERROR WHILE WRITING TRACES',
     +' PRIOR  TO DIP FILTERING.',/,
     +' REWIND I/P TAPE TO START OF NEXT OUTPUT CDP')
      GOTO 510
C
 41   WRITE(2,2041)
      TYPE 2041
 2041 FORMAT(' DISK READ ERROR WHILE READING TRACES',
     +' FOR VELOCITY FILTERING OR AUTOSTATICS.',/,
     +' REWIND BOTH I/P AND O/P TAPES TO START OF CURRENT CDP')
      GOTO 510
 42   WRITE(2,2042)
      TYPE 2042
 2042 FORMAT(' DISK WRITE ERROR WHILE WRITING TRACES',
     +' FOR VELOCITY FILTERING OR AUTOSTATICS.',/,
     +' REWIND BOTH I/P AND O/P TAPES TO START OF CURRENT CDP')
      GOTO 510
C
 51   WRITE(2,2051)
      TYPE 2051
 2051 FORMAT(' DISK READ ERROR WHILE READING TRACES',
     +' FOR DIP FILTERING.',/,
     +' REWIND I/P TAPE TO START OF NEXT OUTPUT CDP')
      GOTO 510
C
 500  WRITE(2,2500)
      TYPE 2500
 2500 FORMAT(///,' ***** NORMAL TERMINATION *****')
C
 510  WRITE(2,2510)(FTRCNO-1)
 2510 FORMAT(///,' LAST OUTPUT TRACE SEQUENCE NUMBER IS:',F15.0)
      WRITE(2,2520) ICDPNO
```

```
 2520 FORMAT(///,' CDP NUMBER OF LAST GATHER WRITTEN:',I5)
      IF(ISTOP.NE.0) WRITE(2,2530)
 2530 FORMAT(///,' * WARNING * ERROR EXIT',/,
     +' CHECK POSITION OF BOTH READ AND WRITE TAPE DRIVES',/,
     +' ALSO CHECK LAST TRACE SEQUENCE NUMBER AND CMP NUMBER')
C
C  CLOSE DOWN CODE
C
      CLOSE(UNIT=2)
      CALL CLOSEC(IDPRT)
      CALL IFREEC(IDPRT)
      CALL CLOSEC(IVFRT)
      CALL IFREEC(IVFRT)
      CALL CLOSEC(10)
      CALL CLOSEC(11)
      CALL CLOSEC(12)
      CALL CLOSEC(13)
      CALL CLOSEC(14)
      RETURN
      END




      SUBROUTINE DEPTH(IDEPTH,IORFNO,IORFTR,IDCDP,IDS,IDR)
C
      VIRTUAL IDEPTH(10000)
C
      COMMON/CONST/ NSAMP,NSAMP2,FSAMP
      COMMON/STATDP/IDFLG,IDPTOT,IDPINT,ISSHOT,IXES,IXSR,IXSS,IXRR
C
      DO 100 I=1,3
      IF(I.EQ.1) P=FLOAT(IORFNO-(IXES+((IXSR+(IORFTR-1)*IXRR)/2))/IXSS)
      IF(I.EQ.2) P=FLOAT(IORFNO-(IXES/IXSS))
      IF(I.EQ.3) P=FLOAT(IORFNO-(IXES+IXSR+(IORFTR-1)*IXRR)/IXSS)
      P=P-FLOAT(ISSHOT)+1.0
      IF(P.LT.1.0) P=1.0
      L=IFIX((P-1.0)/FLOAT(IDPINT))+1
 10   IF(L.LE.(IDPTOT-1)) GOTO 20
      L=IDPTOT-1
      IF(P.GT.FLOAT((IDPTOT-1)*IDPINT+1)) P=FLOAT((IDPTOT-1)*IDPINT+1)
 20   D=FLOAT(IDEPTH(L))+FLOAT(IDEPTH(L+1)-IDEPTH(L))*
     +(P-1.0-FLOAT((L-1)*IDPINT))/FLOAT(IDPINT)
C
C  CHANGE DEPTH FROM METRES TO ABSOLUTE ONE-WAY SAMPLES
C
      ID=IFIX(D*FSAMP/1.5)
      IF(I.EQ.1) IDCDP=ID
      IF(I.EQ.2) IDS=ID
      IF(I.EQ.3) IDR=ID
 100  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE OPERVF(RESULT,NOP,IADR,IVELSH)
C
C  SUBROUTINE TO GENERATE VELOCITY FILTER OPERATOR
C
      VIRTUAL RESULT(32)
C
      CALL APWAIT
      CONST=-(2.*3.1416*3.1416)
      IF(IVELSH.GT.100) CONST=-CONST
      M2=(NOP-1)/2
      M1=-M2
      JADR=IADR
C
      DO 100 I=M1,M2
      II=I
      L=1
      DO 200 J=M1,M2
      JJ=J
      IF(II.NE.0.AND.JJ.NE.0) GOTO 150
      RES=0.0
      IF(II.EQ.0.AND.JJ.EQ.0) RES=0.5
      GOTO 180
  150 FACTOR=4.0
      IF(MOD(II,2).EQ.0.OR.MOD(JJ,2).EQ.0) FACTOR=0.0
      RES=FACTOR/(CONST*II*JJ)
  180 RESULT(L)=RES
      L=L+1
  200 CONTINUE
      CALL APWAIT
      CALL APPTXM(RESULT(1),JADR,NOP,2)
      CALL APWAIT
      JADR=JADR+NOP
  100 CONTINUE
      RETURN
      END
```

```
;
; ROUTINE DATE
;
; A.N. BOWEN SEP 84
;
; ROUTINE USES PROGRAMMED REQUEST .DATE TO OBTAIN CURRENT DATE
;--------
; CALL FORMAT:
;
; CALL DATE(IDAY,IMONTH,IYEAR)
; IDAY   -  DAY NUMBER (1-31)
; IMONTH -  MONTH NUMBER (1-12)
; IYEAR  -  YEAR NUMBER (1900-1999)
; (N.B. ALL SET TO ZERO IF NO DATE SPECIFIED)
;--------
 .GLOBL DATE
 .MCALL .DATE
;
DATE: TST (R5)+     ; SKIP NUMBER OF ARGUMENTS
 .DATE      ; GET SYSTEM DATE
 MOV R0,R2      ; TAKE COPY
 BIC #^C37,R2    ; CLEAR ALL BUT YEAR
 ADD #^D1972,R2    ; ADD OFFSET OF 1972
 ASR R0
 ASR R0     ; PUT MONTH ON BYTE BOUNDARY
 MOV R0,R1      ; RECOPY
 SWAB R1     ; PUT MONTH IN LOW BYTE
 BIC #^C37,R1    ; CLEAR ALL BUT MONTH
 ASR R0
 ASR R0
 ASR R0     ; PUT DAY ON BYTE BOUNDARY
 BIC #^C37,R0    ; CLEAR ALL BUT DAY
 MOV R0,@(R5)+    ; PUT DAY TO IDAY
 MOV R1,@(R5)+    ; PUT MONTH TO IMONTH
 MOV R2,@(R5)    ; PUT YEAR TO IYEAR
 RTS PC     ; RETURN TO FORTRAN
 .END
```

PLOT PROGRAM

1. PLOT PROGRAM OVERVIEW

2. PROGRAM DESCRIPTIONS

    2.1 Control and Input Program, PLOT

        2.1.1 Velocity Panel Data File

        2.1.2 Documentation Panel Data File

    2.2 Frequency Analysis Program, FRQANL

    2.3 Vector to Raster Conversion Program, PLTRSM

    2.4 Data to Raster Image Program, PLTDAT

    2.5 Plot File Output to Plotter Program, PLTOUT

3. PLOT PROGRAM PARAMETER FILE DESCRIPTION - DK6:PLOT.DAT

4. RT-11 DISC DATA FILE DESCRIPTION AND PLOT ARCHIVING

1. PLOT PROGRAM CONCEPTS

The plot program takes a time series data set and produces either a raster image or spectral analysis output on the plotter. The time series data may be stored on tape in Durham SEG-Y or on disc as a RT-11 format file, see section 4 for further details.

The program is actually a composite of five seperate routines that work independantly of each other. After the termination of a particular routine the next is called automatically, chained, without user intervention until the final routine copies the image created to the plotter at which time the program terminates normally. Below are listed the major programs used.

PROGRAM NAME	PARAMETER FILE NAME	DESCRIPTION
PLOT	DK6:PLOT.DAT	Either reads parameter file or runs interactive input to determine plot requirements. Reads data from SEG-Y tape to disc and creates any background needed.
FRQANL		Called if spectral analysis required.
PLTRSM		Takes background or spectral analysis vector files and converts them to a raster image.
PLTDAT		Called when producing a seismic data plot, takes time data and produces raster image, also merges this with any background.
PLTOUT		Copies the spooled disc image file to the plotter.

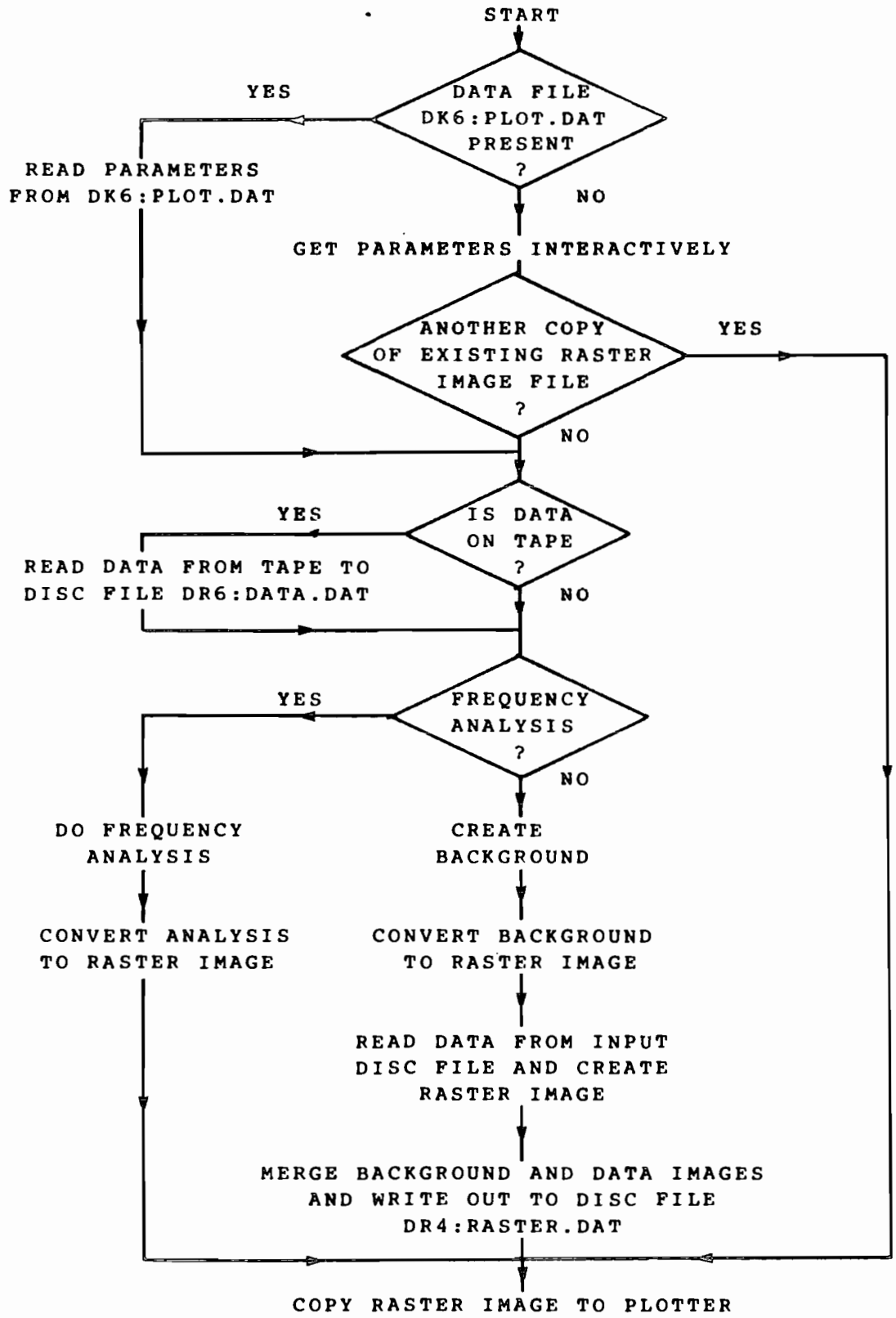
When presenting the seismic data, a pseudo three-dimensional raster image is created of two-way travel time against offset with trace amplitude as the deflection along the time axis. The raster image may be output in two forms. A section plot, this is used for stacked data or constant offset sections. A gather or panel plot, this outputs each shot gather, CMP gather or stacked section panel seperated from each other by five blank records, for easy identification of individual gathers or for a section that has been processed by several different methods, e.g. constant velocity stacks.

For spectral analysis only one trace may be processed at a time giving rise to three output spectra, power, amplitude and phase. The power phase spectrum is not produced as it is always zero for all frequencies.

When plotting from tape the start gather number may be specified, the program will then search from its' current position for the first gather with a number greater than or equal to that given. When plotting from a disc file, the plot always starts at the first gather present, but it is possible to only plot part of the input data if required.

Section 2 explains each program in turn from reading the data from tape to producing the final plot. Included in the first section, 2.1, is a description of the interactive input for running the program suite and the format of optional data files used when velocity analysis information or extra documentation are to be included in the plot. Section 3 examines the control parameter file that may be used for particular plotting requirements instead of the interactive mode.

The flow diagram shows how the input data is processed to give the final output plot. The program is structured so that it may only be run from the control program PLOT, all other routines are chained automatically without user intervention.



Flow Diagram for the Plot Creation

Below are listed the program names and the relevent input and output devices or files.

PROGRAM NAME	PARAMETER FILE NAME	INPUT DATA	OUTPUT DATA
PLOT	DK6:PLOT.DAT	TAPE OR DISC FILE	
	interactive	TAPE OR DISC FILE	
If spectral analysis requested jump to program FRQANL			
If seismic data plot requested create background			DR4:PARM.BIN DR4:VECTR2.BIN
If velocity panel required		DK6:VELRMS.DAT	
If documentation panel required		DK6:DOC.DAT	
If input on tape			DR6:DATA.DAT
If input on disc			disc file name
FRQANL			
If input on tape		DR6:DATA.DAT	DR4:PARM.BIN DR4:VECTR2.BIN
If input on disc		disc file name	DR4:PARM.BIN DR4:VECTR2.BIN
PLTRSM		DR4:PARM.BIN DR4:VECTR2.BIN	DR4:VPLOT.DAT
If spectral anaysis requested jump to program is PLTOUT			
PLTDAT			
If input on tape		DR6:DATA.DAT DR4:VPLOT.DAT	DR4:RASTER.DAT
If input on disc		disc file name DR4:VPLOT.DAT	DR4:RASTER.DAT
PLTOUT			
If spectral analysis requested		DR4:VPLOT.DAT	plotter
If seismic data plot requested		DR4:RASTER.DAT	plotter

2. PROGRAM DESCRIPTIONS

2.1 Control and Input Program, PLOT

The plot program may either run interactively from the terminal or, if present, from a parameter file, DK6:PLOT.DAT. The interactive option is described here in section 2.1, the parameter file is described in section 3.

As the program uses large files it is advisable to ensure that there is enough work space on discs DR4: and DR6: by using the following command to delete any unwanted files

```
DEL DR4:*.*
or  DEL DR6:*.*
```

the program will prompt for each file to be deleted.

To run the program in either mode type

```
RUN DK1:PLOT
```

INTERACTIVE INPUT DESCRIPTION

If the interactive mode is required, then prior to starting the program ensure that there is no parameter file of the name

```
DK6:PLOT.DAT
```

Do this using the command

```
DIR DK6:PLOT.DAT
```

If the specified file is present either delete or rename using one of the following commands

```
DELETE DK6:PLOT.DAT                  to delete

RENAME DK6:PLOT.DAT DK6:PLOT.OLD     to rename file
```

Run the program as above.

The program first prompts for which type of plot is required. There are four options.

PLOT EXISTING IMAGE FILE this will only create another paper copy of a previously created image file. The file may either be stored on disc or as an RT-11 file on a plot tape. The archiving of plots is covered in setion 4.
The final image is better if the plot is taken from disc.
N.B. This program will not work with velocity analysis plots, see relavent program description for multiple copies.

FREQUENCY ANALYSIS this option will produce the power, amplitude and phase spectra for a given trace.

GATHER OR PANEL PLOT this produces short sections of each gather or panel input separated by five blank traces for easy identification.

SECTION PLOT this produces a continuous plot with out any spaces except for zero or dead traces.

The program then prompts for the input data device.

GIVE TAPE DRIVE NUMBER, -1 FOR DISC FILE a reply of 0,1 or 2 indicates which tape drive has the Durham SEG-Y data tape mounted on from which the plot is to be taken.

-1 specifies the input data to plot is already stored in a RT-11 disc file. A supplimentary question will be asked to identify the name of the file. Give full name including : and ..

Having identified the input data device the program searches for the nearest header block, for the disc file this is block 0, see section 4, for a tape this may either be the binary tape header if the tape is at load point, or the previous trace header if the tape is at a random position. As much information as possible is extracted and displayed. If the parameters are not correct then a (N)O reply will make the program prompt for the information. A (Y)ES reply will skip the questions concerning the found parameters. All the possible questions are listed below.

FIRST CMP NUMBER TO PLOT this is only relavent for tape input. If a number is given, the tape will search from its' current position to the first CMP gather with a number greater or equal to that specified, the CMP number on the plot will represent the first found. If a carriage return is typed then the program assumes that the input data has not been sorted from shot gathers and another question is asked prompting for the first shot gather to plot, again the program will search the input tape for the next shot number greater or equal to that specified.

TOTAL NUMBER OF INPUT TRACES PER GATHER OR PANEL only asked when creating a gather or panel plot, assumed to be 1 for section plots. This represents the number of traces in each input gather when reading the data from tape.

GIVE NUMBER OF SAMPLES PER TRACE this indicates the number of samples to expect in each input trace when reading from tape or disc. An error in this number will cause the tape read routine to fail or give unpredictable results from a disc file input.

SAMPLING FREQENCY the sampling spectral of the input data in kilohertz.

Plot Parameters.

DELAY TO START OF PLOT this determines the start time of the plotted section or the start time of the data input to the spectral analysis routine. In milliseconds.

-1 the start time of the plot is the same as for the input data.

PLOT LENGTH MILLISECONDS this number specifies the length of either the plot or the spectral analysis. In milliseconds.
N.B. the end time equals the start time plus the plot length.

-1 the plot length is the same as the trace length.

When the input data is on tape, to save disc file space only the window of data requested is copied. The appropriate numbers are set in the disc header block. See section 4.

Checks are now made that the generated data file will fit on the DR6: disc platter.

GIVE NUMBER OF GATHERS OR PANELS TO PLOT this determines how many gathers to include in the final plot. For disc files this number may not exceed the length of the input data file.

GIVE FIRST AND LAST TRACE NUMBERS TO INCLUDE IN PLOT this allows the windowing of the input gather when read from tape, to only include the trace sequence numbers specified, e.g. by entering 1,1 the first trace will be stripped off and presented as a constant offset section. A carriage return includes all input traces.

GIVE NUMBER OF HORIZONTALLY SUMMED TRACES this question is only asked for section plots. If while post stack processing a horizontal sum is performed then this number is used to modify the plotted CMP numbers to correct for the contraction of the data set.

Checks are now made that the generated plot files will fit on the DR4: disc platter.

If doing a spectral analysis only one gather is read from tape or only the first gather from a disc file is used. From that gather an individual trace must be selected:

GIVE TRACE NUMBER FOR ANALYSIS

Display Options.

The next section deals with the way in which the data is to be presented. This section is not relevant for spectral analysis.

DISPLAY TYPES -2,-1,0,1,2 indicate which display mode required, for normal seismic plots with traces spacing of greater than 0.05 inches options -1,0 or 1 may be used. If the trace spacing is less than 0.05 inches then options -2 or 2 give clearer results.

APPLY EXTRA A.G.C. FUNCTION for raw data sets it may be necessary to apply an automatic gain control function so that events can be seen lower in the section. This option crudely provides for this need without using the Time Sequence Analysis package. It is a fixed function and may be used on any data set except autocorrelation functions.

PLOT GAIN FUNCTION this is a fixed gain applied to all plotted traces prior to rasterization. The gain should be such that the deflection of the peak arrival should be equal to twice the trace spacing, in inches. If this is not correct another plot may be generated with a different gain. If an A.G.C. function has been applied within this program the gain may be defaulted by typing carriage return.

PLOT LENGTH INCHES this determines the vertical scaling of the trace as
plot length (millisec) / plot length (inches)
The length may be upto 10.24 inches long, if a number greater than 10.24 is entered or the entry is defaulted by a carriage return then a value of 10.24 is taken.

TRACE SEPARATION INCHES this determines the horizontal scaling of the plot. This number may be defaulted by typing carriage return to a value of 0.1 inches.

After applying the gain function a clip is used to limit the maximum amplitude in the plot. For normal wiggly line plots the clip levels are set to plus and minus twice the trace spacing. If either of the clipped display modes has been selected, options 2 or -2, then only one polarity of the signal is plotted. For option number 2 the plot clip levels are 0 to twice the trace spacing as above, the lower limit may be increased to include less of the input data as required, typically numbers in the range of 0.005 to 0.01 inches are used, as this will remove low amplitude noise, each raster plotted has a width of 0.005 inches. The lower clip parameter is determined by a supplimentary question:

GIVE MINIMUM ABSOLUTE VALUE TO PLOT IN INCHES.

For option -2 the operation is the same as above except the clip levels now work between 0 and minus twice the trace spacing.

REVERSE PLOT reverses the order in which the traces are plotted.

INCLUDE DEAD TRACES this allows 'dead' traces, i.e. those with a SEG-Y type greater than 2, to be treated as good traces. After the process of dip filtering or migration even 'dead' traces contain data smoothed out by either of the above processes. This allows that data to be plotted.

Plot Documentation

On all plots there is a minimum documentation panel this includes a title and a line number. With a maximum background more details may be included as given below.

Minimum documentation

GIVE LINE NUMBER	line identification number, integer number only.
PLOT TITLE	an ASCII title not more than 20 characters long.

Maximum documentation

This automatically surrounds the plot with a border and a time scale on both the left and right hand sides of the plot. The top border has numbers showing the CMP or shot number. Either or both of the following information may be added:

VELOCITY ANALYSIS	to the right of the plot for section data the velocity analysis data may be added. A '*' is marked on the plot as an indication of each analysis position. The data is taken from the file DK6:VELRMS.DAT, the structure of this file is given in section 2.1.1.
FULL DOCUMENTATION	this adds a large header block that will contain the text from the file DK6:DOC.DAT, structure of this file is given in section 2.1.2.

2.1.1 Velocity Data File

The velocity data file structure is the same as that used by the Time Sequence Analysis package for the r.m.s. stacking velocities, this means that the same file may be used directly. The program automatically scans the file for those analysis that fall within the plot window and these are extracted for presentation on the final section.

The velocity data file has the following structure:

```
VELOCITY ANALYSIS 1
VELOCITY ANALYSIS 2
    .       .
    .       .
```

Each velocity analysis is entered in CMP order an have the following form:

```
CMP NUMBER,  NUMBER OF LAYERS
TIME(1),  VELOCITY(1)
TIME(2),  VELOCITY(2)
  .          .
  .          .
```

The time represents the two-way travel time in milliseconds and the velocity is in kilometres/second. For futher details see the Time Sequence Analysis Package documentation section 4.

2.1.2 Documentation Panel Data File

The documentation to be appended to the final plot is written in the data file DK6:DOC.DAT. The lay out in this file represents how the documentation panel will be presented. Each line may contain upto 80 characters with upto 50 lines in the file. The size of the documentation panel is such that a Durham SEG-Y ASCII tape header may be output. To achieve this do a short run of the processing package to a scratch tape starting with the input tape at load point, this will copy the tape headers into a file DK6:SEISHD.DAT copy this file to DK6:DOC.DAT and use the RT-11 Edit program to remove the binary header information.

Program Operation

If the data is to be input from tape, the program copies the data to the disc file DR6:DATA.DAT. For spectral analysis control is immediately passed to the frequency analysis program, see section 2.2. Otherwise the background for the seismic data plot is created including velocity information and extra documentation as required. Control is then passed to the vector to raster conversion routine PLTRSM.

2.2 Frequency Analysis Program, FRQANL

This routine is called when a spectral analysis is requested, this option may not be called from the disc parameter file DK6:PLOT.DAT. The routine performs the spectral analysis on one trace taken from the disc data file. The trace is read from disc time windowed and passed into the array processor. A ten sample taper is applied to the start and end of the data trace to avoid discontinuities occuring within the fast fourier transform routine. After transformation the power spectrum is extracted as the product of the complex vector and its' complex conjugate. The amplitude and phase spectra are computed from the complex vector and the phase spectrum is un-wrapped, as the tan fuction used is only defined between plus and minus pi radians. All the plots are automatically scaled to fit the plot space available and output as vector plot files.

The program then calls the plot vector to raster convertion routine PLTRSM.

2.3 Vector to Raster Conversion Routine, PLTRSM

This is a modified version of the Versatec software routine RASM. It reads the plot vector files DR4:VECTR2.BIN and DR4:PARM.BIN and outputs a raster image file DR4:VPLOT.DAT.

This routine is always called, either to convert the background for a seismic plot or to convert a spectral analysis plot. For the latter case program control is passed directly to the output routine, PLTOUT.

2.4 Data to Raster Image Program, PLTDAT

This program is used for the conversion of seismic data to a raster image file at the same time merging any background necesary for the PLTRSM output file DR4:VPLOT.DAT and writing the output file DR4:RASTER.DAT.

Initially the minimum plot header is copied from the background file to the output file. When this is complete each trace is read from the input data disc file; windowed to fit the requested start and end times; the automatic gain function is applied if necessary; the whole trace is multiplied by the fixed gain factor; re-sampled to the correct time scale and stored in a trace buffer in the array processor. Upto five traces may be stored at any given time allowing the maximum deflection of any individual trace to be twice the trace spacing. The program outputs one trace spacing's worth of rasters, moves the trace buffer along one trace and inputs the next seismic data trace from the input file. A separate background buffer is refilled as necessary from the DR4:VPLOT.DAT file, the final raster output to DR4:RASTER.DAT being the logically merged product of the background and the data image.

2.5 Plot File Output to Plotter Program, PLTOUT

This program takes either the DR4:VPLOT.DAT file output by the spectral analysis or the DR4:RASTER.DAT file from the seismic data plot routine and copies to the plotter. At the end of this routine the program suite terminates and control is returned to the user.

3. Plot Parameter File - DK6:PLOT.DAT

The plot program may be run from a parameter file, provided that only section or gather plots are required. The file has the following structure:

```
PLOT TYPE
IGATHR,ITRCIN,NSMPIN,FSAMP

either TPDRR,IDEAD

or     FULL DISC FILE NAME

ISHOT,ISTART,NDELAY,NLENG,NHSUM,ITS,ITE
IPOL,IAGC,IREV,IBACK,IVEL,IDOC,NANNO
GAIN,TLENG,TSPACE,CLEV
LINE,TITLE
```

PLOT TYPE — enter control word in block capitals:

SECTION for a section plot.
GATHER for a gather plot.

Input parameters

IGATHR	(I6)	the number of gathers to plot.
ITRCIN	(I6)	the number of traces per input gather, set to 1 for section plot.
NSMPIN	(I6)	the number of samples per input trace.
FSAMP	(F10.0)	the sampling frequeny in kilohertz.
TPDRR	(I6)	the drive number with the input tape mounted and loaded.
IDEAD	(I6)	only used with tape input to ignore SEG-Y type numbers 0 = zero all dead traces on input. 1 = include any data in dead traces in plot.
FULL FILE NAME		the full disc file name of the input data.

Plot parameters

ISHOT	(I6)	0 = CMP data input. 1 = shot data input.
ISTART	(I6)	start gather number, either shot or CMP depending on the value of ISHOT, to start plot. This number is may be updated when the input data is on disc to the number of the first gather present.
NDELAY	(I6)	the delay to the start of the plot in milliseconds.
NLENG	(I6)	the length of the plot in milliseconds.
NHSUM	(I6)	this represents the number of stacked traces added together as part of post stack processing, option 23 in the Time Sequence Analysis package. If not required this value must be set to 1.
ITS	(I6)	when using GATHER mode this number represents the first trace in a gather to plot. For SECTION plots this field may be left blank.
ITE	(I6)	when using GATHER mode this number represents the last trace in a gather to plot. For SECTION plots this field may be left blank.
IPOL	(I6)	plot option: -2 = shaded negative lobes only. -1 = as above with full wiggly line. 0 = unshaded wiggly line only. 1=shaded positive lobe and full wiggly line. 2 = shaded positive lobe only.
IAGC	(I6)	automatic gain control request 0 = no a.g.c. 1 = with a.g.c.
IREV	(I6)	plot sense 0 = plot first trace on left hand side of image. 1= plot first trace on right hand side of image.

IBACK	(I6)	background request 0 = minimum background 1 = maximum background

The next three parameters need only be included if the maximum background is requested.

IVEL	(I6)	velocity panel request 0 = no velocity panel 1=include velocity panel, velocity data contained in DK6:VELRMS.DAT
IDOC	(I6)	documentation panel request 0 = no documentation panel 1= include documentation panel, use data contained in DK6:DOC.DAT
NANNO	(I6)	annotation interval for section plots only, 0 or blank field causes default annotation interval dependent on trace spacing and horizontal sumnation factors.
GAIN	(F10.0)	plot gain.
TLENG	(F10.0)	plot length in inches, if zero defaults to 10.24 inches.
TSPACE	(F10.0)	trace spacing in inches, if zero defaults to 0.1 inches.
CLEV	(F10.0)	clip level in inches, only used if the plot option has a magnitude of 2. Clip level increases in units of 0.005 inches.
LINE	(I6)	line number for plot title.
TITLE	(20A1)	twenty character string for plot title.

4. RT-11 DISC DATA FILE DESCRIPTION AND PLOT ARCHIVING

On transfer of data from tape to disc, the program strips off the SEG-Y trace headers, takes the input samples that lie within the requested time window, and rounds up the sample count to a multiple of 128, a disk block boundary, by padding the input traces with zero samples. The traces are then packed together into a disc file starting at block 1, the second block in the file, without individual trace headers. The first block, block 0, of the file contains the trace header from the first trace read. The constants in the file header are updated to reflect any change in the number of samples or the number of traces per input gather that may have occurred during transfer.

A detailed description of the entries in the file header may be found in the Time Sequence Analysis Package documentation section 6.

The total size of the RT-11 disc data file created when reading from tape may be calculated as follows:

Time window = End time - Start time

No of samples = Time window * Sampling frequency

No of data blocks = Truncate to integer ((No of samples /128.) + 0.5)

Total no of blocks = (No of data block * total number of traces to plot) + 1

Plot Archiving

To archive plot image files on tape follow the proceedure outlined below:

For a new tape mount a scratch tape on drive 0 and type:

INIT/VOL MT0:

The routine will prompt for a volume identification and an owners name, respond with character strings of not more than 6 letters for the i.d and 10 letters for the name.

e.g. PLOTS and R.W.HOBBS

For an old tape just mount onto drive 0.

To archive a seismic data plot type:

COP DR4:RASTER.DAT MT0:LINE6.PLT

the second name may be any permitted character string provided the name has not already been used on this paticular tape.

Similarly for spectral analysis plots type:

COP DR4:VPLOT.DAT MT0:LINE6.FRQ

again the above condition exists.

To restore plots from tape, again load archive tape onto drive 0. The plot program allows the coping of data directly from tape to plotter. When running the PLOT program use option 'P' and in reply to the file name respond with the full tape file name, e.g.

MT0:LINE6.PLT

the program will search the tape for the specified name. Unfortunately as the tape drive is slower than the plotter a plot taken from tape will have grey lines on every time the plotter waits for the next read to be complete. To overcome this problem first copy the plot file from tape to disc using the reverse of the above copy command.

COP MT0:LINE6.PLT DR4:TEMP.PLT

Again the second name may be chosen as required, provided this name is used when in the plot program. To avoid disc corruption ensure the destination disc has enough contiguous space for the file to be read by using the command:

DIR/FREE DR4:

If in doubt try squeezing the disc, command:

```
SQ/NOQ DR4:
```

If still not enough space can be generated delete old files:

```
DEL DR4:*.*
```

The program will query each file to be deleted a `Y' response will destroy the file. Then squeeze the disc again.

```
C
C   PLOT.FOR.....master control and input program for seismic plotting suite
C
C   This program performs four functions:
C      i)  determines plotting parameters either interactively or from a
C          parameter file DK6:PLOT.DAT
C      ii) if the data to be plotted is stored on tape, it is transferred
C          to a disc file DR6:DATA.DAT for plotting. This allows the plot
C          output trace order to be reversed and time windowing of the data
C      iii)creates the necessary background to merge with final plot.
C      iv) enables the output of extra copies as required
C
C   Having finished the routine chains to the image creation routines,
C   FRQANL for frequency analysis, PLTRSM modified VERSATEC RASM routine
C   to convert background to raster format or PLTOUT for extra copies.
C   To convert the seismic data to raster format and merge the background
C   PLTRSM calls PLTDAT
C   Parameters are passed in a chain block PARM.
C
C    COMMON PARAMETERS
C
C    SEIS        virtual tranfer array for seismic data
C    FSPECR      disc file name for nput stored data default DR6:DATA.DAT
C    FSPECW      disc file name of stored data passed to next routine
C    FRQANL      chain name of frequency analysis program
C    PLTDAT       chain name of raster convertion routine
C    PLTOUT      chain name of plot output routine
C    TMNAM       radix 50 tape drive names
C    PARM        chained parameter list
C    TPDRR
C    TPDRW       tape control words, TPDRW not used
C    ITMERR
C    OUTBUF      logiacal array for the SEG-Y trace header
C    TITLE       title of plot
C
C    PASS        common block for chain parameter array
C    TAPE        common block for tape control
C    DATBUF      common block for trace header information
C    PPEP1       common block for VERSATEC PLOTS call, see VERSATEC doc.
C    CNTRL       common block for control in this program
C    NHSUM       numberof horizontally summed traces
C    IBACK       full background flag 1=on 0=off
C    IVEL        if IBACK=1 then if IVEL=1 velocity information will be
C                included in plot, data read from file DK6:VELRSM.DAT
C    IDOC        if IBACK=1 then if IDOC=1 full documentation panel will
C                be included in plot, text read from file DK6:DOC.DAT
C    NANNO       if IBACK=1 then this determines the annotation interval
C                along the top of the plot. 0 sets a default value
C    IDEAD       if set to 0 all dead traces, SEG-Y i.d. code greater or
C                equal to 2, will be set to zero on plot, if set to 1 all
C                dead traces will be treated as good data, e.g. after dip
C                filtering.
C
C    FSPECP      disc file name of raster plot file used for extra copies
C    IPLOT       plot type flag     1=section
C                                   0=gather
C                                  -1=frequency analysis
C                                -999=extra copy
C    ISHOT       input data type 0=CMP, 1=common shot
C    ISTART      first gather number to be plotted
C    IGATHR      number of gathers to plot
C    ITRCIN      number of traces per input gather
C    NSMPIN      number of samples per input trace
C    NDELAY      delay to start of plot in milliseconds
C    NLENG       length of plot in milliseconds
C    TLENG       length of plot in inches
C    FSAMP       sampling frequency in kHz
C    IAGC        extra AGC flag 0=no, 1=yes
C    NIBS        trace spacing in NIBS, 200 NIBS = 1 inch, calculated
C                from TSPACE in inches
C    GAIN        gain factor to be applied to trace before plotting,
C                when IAGC=1 may be defaulted by entering <CR>
```

```
C     ICLIP       allows the plotting of only +ve or -ve lobes
C     NCLP        if ICLIP=1 this sets minimum level in NIBS to plot,
C                 calculated from clip level CLEV
C     IPOL        plot polarity +ve shaded +ve lobes, -ve shades -ve lobes
C     IREV        reverses output plot sense
C     ISDLY       start delay on output plot in NIBS
C     ISAMPS      time scale in NIBS
C     ICHAN       numberof input gather to input to frequency analysis
C     LINE        line number for plot identification
C     ITS         first trace number to plot
C     ITE         last trace number to plot
C
C   WRITTEN R.W.HOBBS 1985
C
      VIRTUAL SEIS(4156)
      REAL*8 FSPECR,FSPECW
      REAL*4 FRQANL(2),PLTRSM(2),PLTOUT(2),TMNAM(3)
      INTEGER*2 PARM(50),TPDRR,TPDRW,ITMERR(2)
      LOGICAL*1 OUTBUF(240),TITLE(20)
C
      COMMON/PASS/ PARM
      COMMON/TAPE/ TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/DATBUF/ OUTBUF
      COMMON /PPEP1/ IX1,IY1,IX2,IY2,ISCAN,NSCAN,NBAND,NPIS,NIP0,
     +NIPM1,LYNES,NIBSX,MSGLVL,XDOTS,YDOTS,PREF(2),
     +RORG(2),PORT(2,2),INND(4),ALMT,FACT,JPEN,XOFF,
     +XFAC,YOFF,YFAC,NBITS,NBITM1,NBYTES,NBYTM1,MSK,LMSK
      COMMON/CNTRL/ NHSUM,IBACK,IVEL,IDOC,NANNO,IDEAD
C
      EQUIVALENCE (PARM(5),IPLOT),(PARM(6),ISHOT),
     +(PARM(7),ISTART),(PARM(8),IGATHR),(PARM(9),ITRCIN),
     +(PARM(10),NSMPIN),(PARM(11),NDELAY),(PARM(12),NLENG),
     +(PARM(13),TLENG),(PARM(15),FSAMP),(PARM(17),IAGC),
     +(PARM(18),NIBS),(PARM(19),GAIN),(PARM(21),ICLIP),
     +(PARM(22),NCLP),(PARM(23),IPOL),(PARM(24),IREV),
     +(PARM(25),ISDLY),(PARM(26),ISAMPS),(PARM(27),ICHAN),
     +(PARM(28),LINE),(PARM(29),TITLE(1)),(PARM(39),FSPECW),
     +(PARM(43),ITS),(PARM(44),ITE)
C
      DATA TMHN/3RTM /,APHN/3RAP /,DRHN/3RDR /
      DATA FSPECR/12RDR6DATA  DAT/
      DATA TMNAM(1)/3RTM4/,TMNAM(2)/3RTM5/,TMNAM(3)/3RTM6/
      DATA FRQANL(1)/6RDK1FRQ/,FRQANL(2)/6RANLSAV/
      DATA PLTRSM(1)/6RDK1PLT/,PLTRSM(2)/6RRSMSAV/
      DATA PLTOUT(1)/6RDK1PLT/,PLTOUT(2)/6ROUTSAV/
C
C   FETCH HANDLERS
C
      IF(IFETCH(TMHN).NE.0) STOP'TM FETCH ERROR'
      IF(IFETCH(DRHN).NE.0) STOP'DR FETCH ERROR'
      IF(IFETCH(APHN).NE.0) STOP'AP FETCH ERROR'
C
C   CHECK FOR DK6:PLOT.DAT OTHERWISE GO INTERACTIVE
C
      OPEN(UNIT=1,NAME='DK6:PLOT.DAT',TYPE='OLD',ERR=50)
      CALL INPLDK(FSPECR)
      CLOSE(UNIT=1)
      GOTO 60
C
C   GET INTERACTIVE INPUT
C
 50   CALL INPLTT(FSPECR)
      IF(IPLOT.EQ.-999) GOTO 100
C
C   CALL TAPE INPUT IF REQUIRED
C
 60   CALL APINIT(0,0,IAP)
      IF(TPDRR.GE.0) CALL TAPIN(SEIS,FSPECR)
      CALL APRLSE
C
C   MUST UNLOAD AP BEFORE INITIALIZING PLOT AS THEY DO NOT MIX,
C   BOTH USE RT-11 CHANNELS 8 AND 9
```

```
C  PLOT BACKGROUND FOR RASTER IMAGE IF REQUIRED
C
      IF(IPLOT.LT.0) GOTO 100
      CALL PLOTS(0,0,0)
      CALL BACKGD(FSPECR)
      CALL PLOT(0.,0.,999)
 100  CONTINUE
C
C  CLEAR OUT ALL CALLED HANDLERS AND RESET RT-11 I/O LIST
C
      CALL RESET
C
C  DEPENDING ON REQUIRED OPTION CHAIN NEXT ROUTINE
C
      IF(IPLOT.EQ.-999) CALL CHAIN(PLTOUT,PARM,4)
      IF(IPLOT.LT.0) CALL CHAIN(FRQANL,PARM,50)
      IF(IPLOT.GE.0) CALL CHAIN(PLTRSM,PARM,50)
C
C  YOU SHOULD NEVER REACH HERE !!
C
      CALL EXIT
      END




C
C      SUBROUTINE INPLTT
C
C    If the paprmeter file DK6:PLOT.DAT is not found then the program
C    assumes that interactive input is required and will call this routine.
C    Most of the questions should be unambiguous.
C
      SUBROUTINE INPLTT(FSPECR)
C
C  WRITTEN R.W.HOBBS 1985
C
      REAL*8 FSPECR,FSPECW,FSPECP,FSPECI
      REAL*4 TMNAM(3)
      INTEGER*2 PARM(50),TPDRR,TPDRW,ITMERR(2),SPEC(39),EXT(4)
      LOGICAL*1 FBUF(16),TITLE(20),IANS,LERR
C
      COMMON/PASS/ PARM
      COMMON/TAPE/ TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/CNTRL/ NHSUM,IBACK,IVEL,IDOC,NANNO,IDEAD
C
      EQUIVALENCE (PARM(1),FSPECP),(PARM(5),IPLOT),(PARM(6),ISHOT),
     +(PARM(7),ISTART),(PARM(8),IGATHR),(PARM(9),ITRCIN),
     +(PARM(10),NSMPIN),(PARM(11),NDELAY),(PARM(12),NLENG),
     +(PARM(13),TLENG),(PARM(15),FSAMP),(PARM(17),IAGC),
     +(PARM(18),NIBS),(PARM(19),GAIN),(PARM(21),ICLIP),
     +(PARM(22),NCLP),(PARM(23),IPOL),(PARM(24),IREV),
     +(PARM(25),ISDLY),(PARM(26),ISAMPS),(PARM(27),ICHAN),
     +(PARM(28),LINE),(PARM(29),TITLE(1)),(PARM(39),FSPECW),
     +(PARM(43),ITS),(PARM(44),ITE)
      EQUIVALENCE (SPEC(16),FSPECI)
C
      DATA EXT/4*0/
C
C  SET UP DEFAULT VALUES
C
      IGATHR=1
      IAUTO=0
      ICLIP=0
      IAGC=0
      IBACK=0
      IVEL=0
      IDOC=0
      IREV=0
      IDEAD=0
      ISHOT=0
```

```
C
C  INPUT PARAMETERS
C
      TYPE 1000
 1000 FORMAT(' ** Plotting Program **'//)
C
C  DETERMINE WHAT TYPE OF PLOT IS REQUIRED
C
 110  TYPE 1100
 1100 FORMAT(' Plot Options',/,
     +' (P)lot existing raster image file',/,
     +' (F)requency analysis',/,
     +' (G)ather plot or panel plot',/,
     +' (S)ection plot',/,
     +'$       Give option letter (P/F/G/S) ? ')
      ACCEPT 2020,IANS
      IF(IANS.NE.'P'.AND.IANS.NE.'F'.AND.IANS.NE.'S'
     +.AND.IANS.NE.'G') GOTO 110
 2000 FORMAT(12I6)
 2010 FORMAT(6F10.0)
 2020 FORMAT(A1)
      IF(IANS.EQ.'P') IPLOT=-999
      IF(IPLOT.EQ.-999) GOTO 500
      IF(IANS.EQ.'S') IPLOT=1
      IF(IANS.EQ.'G') IPLOT=0
      IF(IANS.EQ.'F') IPLOT=-1
      TYPE 1200
 1200 FORMAT(/,' Input Parameters',/)
      TYPE 1210
 1210 FORMAT('$Give input tape drive no., -1 for disk file',
     +' (integer) ? ')
      ACCEPT 2000,TPDRR
      IF(TPDRR.LT.0) GOTO 122
C
C  READ TAPE HEADER OR NEAREST TRACE HEADER TO DETERMINE INPUT DATA
C
      FSPECW=FSPECR
      CALL TAPHBK(ITRCIN,NSMPIN,NHSUM,FSAMP)
      TYPE 1232
 1232 FORMAT(/)
      IF(ITRCIN.GT.0) TYPE 1233,ITRCIN
 1233 FORMAT(' NUMBER OF TRACES PER GATHER = ',I4)
      IF(NSMPIN.GT.0) TYPE 1230,NSMPIN
 1230 FORMAT(' NUMBER OF SAMPLES PER TRACE = ',I4)
      IF(FSAMP.GT.0.0) TYPE 1231,FSAMP
 1231 FORMAT(' SAMPLING FREQUENCY = ',F5.3,' KHZ')
      IF(NHSUM.GT.1) TYPE 1235,NHSUM
 1235 FORMAT(' HORIZONTAL SUM OF',I3,' STACKED TRACES')
      TYPE 1234
 1234 FORMAT('$O.K. (Y/N) ? ')
      ACCEPT 2020,IANS
      TYPE 1232
      IF(IANS.EQ.'Y') GOTO 130
      ITRCIN=0
      NSMPIN=0
      FSAMP=0.0
      NHSUM=0
      GOTO 130
C
C  IF INPUT FROM DISC GET FILE NAME
C
 122  TYPE 1220
 1220 FORMAT('$Give full input data file name ? ')
      CALL GETSTR(5,FBUF,14,IERR)
      IF(IERR.NE..FALSE.) STOP'ILLEGAL FILE NAME'
      IF(ICSI(SPEC,EXT,FBUF,,0).NE.0) STOP'FILE NAME ERROR'
      FSPECR=FSPECI
      FSPECW=FSPECR
C
C  READ DISC HEADER BLOCK TO GET INPUT PARAMETERS
C
      CALL DSCHBK(FSPECR,IAUTO,NGATHR,ISTART,ITRCIN,ISHOT,
```

```
     +              NSMPIN,NHSUM,FSAMP)
      TYPE 1232
      IF(IAUTO.EQ.1) TYPE 1240
 1240 FORMAT(' AUTOCORRELATION DATA FILE')
      IF(ISHOT.EQ.0.AND.ISTART.GT.0) TYPE 1241,ISTART
 1241 FORMAT(' FIRST CMP NUMBER TO PLOT = ',I6)
      IF(ISHOT.GT.0.AND.ISTART.GT.0) TYPE 1242,ISTART
 1242 FORMAT(' FISRT SHOT NUMBER TO PLOT = ',I6)
      IF(NGATHR.GT.0) TYPE 1243,NGATHR
 1243 FORMAT(' TOTAL NUMBER OF GATHERS IN FILE = ',I6)
      IF(ITRCIN.GT.0) TYPE 1233,ITRCIN
      IF(NSMPIN.GT.0) TYPE 1230,NSMPIN
      IF(FSAMP.GT.0.0) TYPE 1231,FSAMP
      IF(NHSUM.GT.1) TYPE 1235,NHSUM
      TYPE 1234
      ACCEPT 2020,IANS
      TYPE 1232
      IF(IANS.EQ.'Y') GOTO 132
      ITRCIN=0
      ISHOT=0
      NSMPIN=0
      FSAMP=0.0
      NGATHR=0
      NHSUM=0
      GOTO 132
C
C  GET PLOT LIMITS
C
 130  IF(ISTART.GT.0) GOTO 132
      TYPE 1300
 1300 FORMAT('$Give first CMP number to plot, <CR> for shot data',
     +' (integer) ? ')
      ACCEPT 2000,ISTART
      IF(ISTART.GT.0) GOTO 132
      ISHOT=1
      TYPE 1310
 1310 FORMAT('$Give first shot number to plot (integer) ? ')
      ACCEPT 2000,ISTART
 132  IF(ITRCIN.GT.0) GOTO 133
      TYPE 1320
 1320 FORMAT('$Give total number of traces per input',
     +' gather or panel (integer) ? ')
      ACCEPT 2000,ITRCIN
 133  IF(NSMPIN.GT.0) GOTO 134
      TYPE 1330
 1330 FORMAT('$Give number of samples per input trace (integer) ? ')
      ACCEPT 2000,NSMPIN
 134  IF(FSAMP.GT.0.0) GOTO 135
      TYPE 1340
 1340 FORMAT('$Give input sampling frequency kHz (real) ? ')
      ACCEPT 2010,FSAMP
 135  TYPE 1350
 1350 FORMAT(/,' Plot parameters',/,
     +' Give delay time to start of plot,',/,
     +'$-1 for delay as in first trace header millisec (integer) ? ')
      ACCEPT 2000,NDELAY
      TYPE 1360
 1360 FORMAT(' Give length of plot,',/,
     +'$-1 for as in first trace header millisec (integer) ? ')
      ACCEPT 2000,NLENG
      IF(NLENG.LT.0) NLENG=IFIX(FLOAT(NSMPIN)/FSAMP)
      IF(IPLOT.LT.0) GOTO 140
      TYPE 1370
 1370 FORMAT('$Give number of gathers or panels to plot (integer) ? ')
      ACCEPT 2000,IGATHR
      IF(ITRCIN.EQ.1) GOTO 138
      TYPE 1380
 1380 FORMAT(' Give first and last trace number(s) in input',
     +' gather to include in plot',/,'$<CR> for all (integer) ?,? ')
      ACCEPT 2000,ITS,ITE
 138  IF(ITS.EQ.0) ITS=1
      IF(ITE.EQ.0) ITE=ITRCIN
```

```
      IF(ITRCIN.GT.1.OR.IPLOT.EQ.0.OR.NHSUM.GT.0) GOTO 139
      TYPE 1390
 1390 FORMAT('$Give number of horizontally summed traces in each',
     +' trace plotted (integer) ? ')
      ACCEPT 2000,NHSUM
 139  IF(NHSUM.EQ.0) NHSUM=1
      IF(TPDRR.LT.0) GOTO 141
      NSAMP=IFIX(FLOAT(NLENG)*FSAMP)
      NBLK=NSAMP/128
      IF(MOD(NSAMP,128).NE.0) NBLK=NBLK+1
      RBLK=FLOAT(ITE-ITS+1)*FLOAT(IGATHR)*FLOAT(NBLK)+1.0
      IF(RBLK.LE.32767.0) GOTO 141
      TYPE 1395
 1395 FORMAT(/,' * WARNING *  DISC FILE FOR DATA TOO LARGE',/,
     +                        EITHER REDUCE LENGTH OF PLOT',/,
     +                        OR TOTAL NUMBER OF TRACES',/)
      GOTO 135
C
C  FREQUENCY ANALYSIS
C
 140  ITS=1
      ITE=ITRCIN
      TYPE 1400
 1400 FORMAT('$Give trace number for analysis from specified gather',
     +' (integer) ? ')
      ACCEPT 2000,ICHAN
      GOTO 151
C
C  GET PLOTTING PARAMETERS
C
 141  TYPE 1410
 1410 FORMAT(/,' Display Options',/,
     +' -2   Shaded negative lobe only',/,
     +' -1   Shaded negative lobe and wiggly line',/,
     +'  0   Wiggly line only',/,
     +'  1   Shaded positive lobe and wiggly line',/,
     +'  2   Shaded positive lobe only',/,
     +'$                Give option number ? ')
      ACCEPT 2000,IPOL
      IF(IAUTO.EQ.1) GOTO 143
      TYPE 1420
 1420 FORMAT('$Apply extra 50 sample AGC function (Y/N) ? ')
      ACCEPT 2020,IANS
      IF(IANS.EQ.'Y') IAGC=1
 143  TYPE 1430
 1430 FORMAT('$Give plot gain factor (real) ? ')
      ACCEPT 2010,GAIN
      TYPE 1440
 1440 FORMAT('$Give plot length inches (real) ? ')
      ACCEPT 2010,TLENG
      IF(TLENG.EQ.0.0) TLENG=10.24
      IF(TLENG.LE.10.24) GOTO 145
      TLENG=10.24
      TYPE 1445
 1445 FORMAT(/,' * WARNING * PLOT LENGTH REDUCED TO 10.24 INCHES',/)
 145  TYPE 1450
 1450 FORMAT('$Give trace separation inches (real) ? ')
      ACCEPT 2010,TSPACE
      IF(TSPACE.EQ.0.0) TSPACE=0.1
      NIBS=IFIX((TSPACE*200.0)+0.5)
      IF(GAIN.EQ.0.0.AND.IAUTO.EQ.1) GAIN=2.0*FLOAT(NIBS)
      IF(GAIN.EQ.0.0.AND.IAGC.EQ.1) GAIN=0.5*FLOAT(NIBS)
      PWIDTH=FLOAT(IGATHR)*FLOAT(ITE-ITS+1)*FLOAT(NIBS)
      IF(IPLOT.EQ.0) PWIDTH=PWIDTH+(FLOAT(IGATHR-1)*5.0*FLOAT(NIBS))
      RBLK=((PWIDTH+500.0+(6.0*FLOAT(NIBS)))*132.0/256.0)+2.0
      IF(RBLK.LE.32767.0) GOTO 146
      TYPE 1455
 1455 FORMAT(/,' * WARNING * PLOT IMAGE FILE TOO LARGE',/,
     +                        EITHER REDUCE TRACE SPACING',/,
     +                        OR TOTAL NUMBER OF TRACES',/)
      GOTO 135
 146  IF(IABS(IPOL).LE.1) GOTO 147
```

```
      ICLIP=1
      TYPE 1460
 1460 FORMAT('$Give minimum absolute level to plot after scaling',
     +' inches (real) ? ')
      ACCEPT 2010,CLEV
      NCLP=IFIX((ABS(CLEV)*200.0)+0.5)
 147  TYPE 1470
 1470 FORMAT('$Reverse plot (Y/N) ? ')
      ACCEPT 2020,IANS
      IF(IANS.EQ.'Y') IREV=1
      IF(TPDRR.LT.0) GOTO 150
      TYPE 1480
 1480 FORMAT('$Include dead traces in plot (Y/N) ? ')
      ACCEPT 2020,IANS
      IF(IANS.EQ.'Y') IDEAD=1
 150  TYPE 1500
 1500 FORMAT(/,' Plot Documentation',/,
     +'$Maximum background (Y/N) ? ')
      ACCEPT 2020,IANS
      IF(IANS.EQ.'Y') IBACK=1
 151  TYPE 1510
 1510 FORMAT('$Give line number (integer) ? ')
      ACCEPT 2000,LINE
      TYPE 1520
 1520 FORMAT(' Give plot title')
      TYPE 1530
 1530 FORMAT(' |<START        STOP>|')
      ACCEPT 2040,(TITLE(I),I=1,20)
 2040 FORMAT(20A1)
C
C  IF FREQUENCY ANALYSIS RETURN
C
      IF(IPLOT.LT.0.OR.
     +   IBACK.EQ.0) GOTO 900
C
C  COME HERE FOR MAXIMUM ANNOTATION
C
      IF(PWIDTH.LT.32768.0) GOTO 160
      TYPE 1455
      GOTO 135
 160  IF(IPLOT.EQ.0) GOTO 162
      TYPE 1600
 1600 FORMAT('$Include velocity anaysis information (Y/N) ? ')
      ACCEPT 2020,IANS
      IF(IANS.EQ.'Y') IVEL=1
      IF(IVEL.EQ.1) PWIDTH=PWIDTH+(24.0*200.0)
      IF(IVEL.EQ.1) TYPE 1610
 1610 FORMAT(/,' *** VELOCITY DATA MUST BE IN FILE',
     +' DK6:VELRMS.DAT ***',/)
 162  TYPE 1620
 1620 FORMAT('$Include full documentation panel (Y/N) ? ')
      ACCEPT 2020,IANS
      IF(IANS.EQ.'Y') IDOC=1
      IF(IDOC.EQ.1) PWIDTH=PWIDTH+(10.0*200.0)
      IF(IDOC.EQ.1) TYPE 1630
 1630 FORMAT(/,' *** DOCUMENT PANEL DATA MUST BE IN FILE',
     +' DK6:DOC.DAT ***',/)
      RBLK=PWIDTH*132.0/256.0
      IF(RBLK.LE.32767.0) GOTO 164
      TYPE 1645
 1645 FORMAT(/,' * WARNING * PLOT IMAGE FILE WITH ANNOTATION TOO LARGE',/,
     +          '              EITHER USE MINIMUM BACKGROUND',/,
     +          '              OR REDUCE PLOT SIZE',/)
      TYPE 1500
      ACCEPT 2020,IANS
      IF(IANS.EQ.'Y') GOTO 135
      IBACK=0
      IVEL=0
      IDOC=0
 164  IF(IPLOT.EQ.0) GOTO 900
      TYPE 1640
 1640 FORMAT('$Give horizontol onnotation intervel, <CR> for default',
```

```
     +' (integer) ? ')
      ACCEPT 2000,NANNO
      GOTO 900
C
C  GET NAME OF PRE-EXISTING RASTER IMAGE FILE N.B. THIS MAY BE AN RT-11 TAPE
C
 500  TYPE 5100
 5100 FORMAT('$Give full input raster image file name ? ')
      CALL GETSTR(5,FBUF,14,IERR)
      IF(IERR.NE..FALSE.) STOP'ILLEGAL FILE NAME'
      IF(ICSI(SPEC,EXT,FBUF,,0).NE.0) STOP'FILE NAME ERROR'
      FSPECP=FSPECI
C
C  RETURN
C
 900  RETURN
      END




      SUBROUTINE INPLDK(FSPECR)
C
C  PLOT DATA FILE INPUT ROUTINE
C
C  WRITTEN R.W.HOBBS 1985
C
      REAL*8 FSPECR,FSPECW,FSPECP,FSPECI
      REAL*4 TMNAM(3)
      INTEGER*2 PARM(50),TPDRR,TPDRW,ITMERR(2),SPEC(39),EXT(4)
      LOGICAL*1 FBUF(16),TITLE(20),IANS,LERR
C
      COMMON/PASS/ PARM
      COMMON/TAPE/ TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/CNTRL/ NHSUM,IBACK,IVEL,IDOC,NANNO,IDEAD
C
      EQUIVALENCE (PARM(1),FSPECP),(PARM(5),IPLOT),(PARM(6),ISHOT),
     +(PARM(7),ISTART),(PARM(8),IGATHR),(PARM(9),ITRCIN),
     +(PARM(10),NSMPIN),(PARM(11),NDELAY),(PARM(12),NLENG),
     +(PARM(13),TLENG),(PARM(15),FSAMP),(PARM(17),IAGC),
     +(PARM(18),NIBS),(PARM(19),GAIN),(PARM(21),ICLIP),
     +(PARM(22),NCLP),(PARM(23),IPOL),(PARM(24),IREV),
     +(PARM(25),ISDLY),(PARM(26),ISAMPS),(PARM(27),ICHAN),
     +(PARM(28),LINE),(PARM(29),TITLE(1)),(PARM(39),FSPECW),
     +(PARM(43),ITS),(PARM(44),ITE)
C
      EQUIVALENCE (SPEC(16),FSPECI)
C
      DATA EXT/4*0/
C
C  SET UP DEFAULT VALUES
C
      ICLIP=0
C
C  INPUT PARAMETERS
C
      TYPE 1000
 1000 FORMAT(' ** Plotting Program **'//)
C
C  DETERMINE WHAT TYPE OF PLOT IS REQUIRED
C
      READ(1,2020) IANS
      IF(IANS.NE.'S'.AND.IANS.NE.'G') STOP'ILLEGAL PLOT OPTION'
 2000 FORMAT(1216)
 2010 FORMAT(6F10.0)
 2020 FORMAT(A1)
      IF(IANS.EQ.'S') IPLOT=1
      IF(IANS.EQ.'G') IPLOT=0
      READ(1,2050) IGATHR,ITRCIN,NSMPIN,FSAMP
 2050 FORMAT(3I6,F10.0)
      READ(1,2020) IANS
```

```
      BACKSPACE 1
      IF(IANS.EQ.'D') GOTO 120
      READ(1,2000) TPDRR,IDEAD
      FSPECW=FSPECR
      GOTO 130
 120  TPDRR=-1
      CALL GETSTR(1,FBUF,14,IERR)
      IF(IERR.NE..FALSE.) STOP'ILLEGAL FILE NAME'
      IF(ICSI(SPEC,EXT,FBUF,,0).NE.0) STOP'FILE NAME ERROR'
      FSPECR=FSPECI
      FSPECW=FSPECR
 130  READ(1,2000) ISHOT,ISTART,NDELAY,NLENG,NHSUM,ITS,ITE
      IF(NHSUM.EQ.0) NHSUM=1
      IF(ITS.EQ.0) ITS=1
      IF(ITE.EQ.0) ITE=ITRCIN
      IF(NLENG.LT.0) NLENG=IFIX(FLOAT(NSMPIN)/FSAMP)
      READ(1,2000) IPOL,IAGC,IREV,IBACK,IVEL,IDOC,NANNO
      IF(IABS(IPOL).EQ.2) ICLIP=1
      READ(1,2010) GAIN,TLENG,TSPACE,CLEV
      IF(TLENG.EQ.0.0.OR.TLENG.GT.10.24) TLENG=10.24
      IF(TSPACE.EQ.0.0) TSPACE=0.1
      NIBS=IFIX((TSPACE*200.0)+0.5)
      NCLP=IFIX((ABS(CLEV)*200.0)+0.5)
      READ(1,2060) LINE,(TITLE(I),I=1,20)
 2060 FORMAT(I6,20A1)
      RETURN
      END



      SUBROUTINE TAPHBK(ITRCIN,NSMPIN,NHSUM,FSAMP)
C
C  THIS ROUTINE DETERMINES INPUT PARAMETERS FROM THE TAPE BY
C  EITHER READING THE BINARY TAPE HEADER BLOCK IF THE TAPE IS
C  AT BOT, OR THE NEXT TRACE HEADER
C
      VIRTUAL HBLK(200)
      REAL*4 TMNAM(3)
      INTEGER*2 HBLK,TPDRR,TPDRW,ITMERR(2)
C
      COMMON/TAPE/ TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
C
C  LOOKUP READ TAPE
C
 370  ICHN=TPDRR+10
      ILK=LOOKUP(ICHN,TMNAM(TPDRR+1),-1)
      ISTUS=IPEEK("172520)
      IF((ISTUS.AND."100).GT.0) GOTO 380
      CALL CLOSEC(ICHN)
      TYPE 3000
 3000 FORMAT(' GIVEN READ DRIVE NOT AVAILABLE',/,
     +'$GIVE NEW READ DRIVE NO. ? ')
      ACCEPT 3100,TPDRR
 3100 FORMAT(I2)
      IF(TPDRR.GE.3) STOP'NON-EXISTANT READ DRIVE'
      GOTO 370
C
C  CHECK BOT FLAG
C
 380  IEOTR=0
      IF((ISTUS.AND."40).LE.0) GOTO 400
      CALL ISPFNW("376,ICHN)
C
C  READ BINARY HEADER
C
      IF(IRMTXM(HBLK(1),ICHN,200,ITMERR(1)).LT.0) STOP'BINARY READ ERR'
      ITRCIN=HBLK(7)+HBLK(8)
      IF(HBLK(9).GT.0) FSAMP=1000.0/FLOAT(HBLK(9))
      NSMPIN=HBLK(11)
      NHSUM=0
```

```
      CALL ISPFNW("376,ICHN)
      GOTO 900
C
C  IF TAPE IS NOT AT BOT READ NEXT TRACE HEADER
C
 400  CALL ISPFNW("375,ICHN)
      CALL ISPFNW("376,ICHN)
      IRD=IRMTXM(HBLK(1),ICHN,120,ITMERR(1))
      IF(IRD.LT.0.AND.ITMERR(1).NE.5) STOP'TRACE READ ERR'
      ITRCIN=0
      IF(HBLK(59).GT.0) FSAMP=1000.0/FLOAT(HBLK(59))
      NSMPIN=HBLK(58)
      NHSUM=HBLK(17)
      CALL ISPFNW("375,ICHN)
      CALL ISPFNW("376,ICHN)
C
C  CLOSE CHANNEL AND RETURN
C
 900  CALL CLOSEC(ICHN)
      RETURN
      END



      SUBROUTINE DSCHBK(FSPECR,IAUTO,NGATHR,ISTART,ITRCIN,ISHOT,
     +                  NSMPIN,NHSUM,FSAMP)
C
C  THIS ROUTINE READS THE FIRST BLOCK OF A DISC FILE AND
C  DETERMINES AS MANY INPUT PARAMETERS AS POSSIBLE
C
C  WRITTEN R.W.HOBBS 1985
C
      VIRTUAL HBLK(256)
      REAL*8 FSPECR
      INTEGER*4 JNUM
      INTEGER*2 HBLK,IDUM(2)
C
      EQUIVALENCE (IDUM(1),JNUM)
C
      ICHN=IGETC()
      ILK=LOOKUP(ICHN,FSPECR)
      IF(ILK.LT.0) STOP'NO SUCH DISC FILE'
C
      IF(IRDRXM(HBLK(1),ICHN,256,0,0).NE.256) STOP'HBLK READ ERR'
C
      ITRCIN=HBLK(18)
      NSMPIN=HBLK(58)
      IF(HBLK(59).GT.0) FSAMP=1000.0/FLOAT(HBLK(59))
      NHSUM=HBLK(17)
      IAUTO=0
      IF(HBLK(15).EQ.9) IAUTO=1
      DO 100 I=1,2
 100  IDUM(I)=HBLK(10+I)
      CALL JJCVT(JNUM)
      CALL IJCVT(JNUM,ISTART)
      IF(ISTART.GT.0) GOTO 900
      ISHOT=1
      DO 110 I=1,2
 110  IDUM(I)=HBLK(4+I)
      CALL JJCVT(JNUM)
      CALL IJCVT(JNUM,ISTART)
 900  NTRCIN=ITRCIN
      IF(ITRCIN.EQ.0) NTRCIN=1
      NBLK=NSMPIN/128
      IF(MOD(NSMPIN,128).NE.0) NBLK=NBLK+1
      ILK=ILK-1
      NTRIN=ILK/NBLK
      NGATHR=NTRIN/NTRCIN
C
C  CLOSE FILE AND CLEAR CHANNEL
```

```
C
      CALL CLOSEC(ICHN)
      CALL IFREEC(ICHN)
      RETURN
      END




      SUBROUTINE TAPIN(SEIS,FSPECR)
C
C  THIS IS A TRANSFER SUBROUTINE FROM DUR-SEG-Y TAPE TO A
C  RT-11 DISC FILE, I.E. COMPACT FORM WITH NO INDIVIDUAL TRACE HEADERS
C  BUT A SINGLE FILE HEADER IN BLOCK ZERO WHICH IS TAKEN FROM THE
C  FIRST TRACE READ. ALL SUBSEQUENT TRACES ARE CORRECTED TO THE
C  SAME DELAY TIME. THE 'TYPE' OF THE DATA IS SET TO 1 INDICATING
C  GOOD DATA
C
C  WRITTEN R.W.HOBBS 1985
C
      VIRTUAL SEIS(4156)
      REAL*8 FSPECR
      REAL*4 TMNAM(3),TRHDBF(60)
      INTEGER*4 JFNUM,JTRAS,JCDPNO,JCDPTR
      INTEGER*2 PARM(50),TPDRR,TPDRW,ITMERR(2),HBLK(256)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/PASS/ PARM
      COMMON/TAPE/ TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/DATBUF/ OUTBUF
      COMMON/CNTRL/ NHSUM,IBACK,IVEL,IDOC,NANNO,IDEAD
C
      EQUIVALENCE (PARM(5),IPLOT),(PARM(6),ISHOT),
     +(PARM(7),ISTART),(PARM(8),IGATHR),(PARM(9),ITRCIN),
     +(PARM(10),NSMPIN),(PARM(11),NDELAY),(PARM(12),NLENG),
     +(PARM(13),TLENG),(PARM(15),FSAMP),(PARM(43),ITS),
     +(PARM(44),ITE)

      EQUIVALENCE (HBLK(1),TRHDBF(1))
      EQUIVALENCE (OUTBUF(9),JFNUM),(OUTBUF(13),JTRAS),
     +(OUTBUF(21),JCDPNO),(OUTBUF(25),JCDPTR),(OUTBUF(29),ITYPE),
     +(OUTBUF(109),IDELAY),(OUTBUF(115),NSMP)
C
C  SET UP CONSTANTS
C
      NCHN=IGETC()
      IBLK=1
      NSMPOT=IFIX(FLOAT(NLENG)*FSAMP)
      NBLK=NSMPOT/128
      IF(MOD(NSMPOT,128).NE.0) NBLK=NBLK+1
      IBKTOT=IGATHR*(ITE-ITS+1)*NBLK+1
      IWORDR=NSMPIN*2+120
      IWORDW=NSMPOT*2
C
C  LOOKUP READ TAPE
C
 370  ICHN=TPDRR+10
      ILK=LOOKUP(ICHN,TMNAM(TPDRR+1),-1)
      ISTUS=IPEEK("172520)
      IF((ISTUS.AND."100).GT.0) GOTO 380
      CALL CLOSEC(ICHN)
      TYPE 3000
 3000 FORMAT(' GIVEN READ DRIVE NOT AVAILABLE',/,
     +'$GIVE NEW READ DRIVE NO. ? ')
      ACCEPT 3100,TPDRR
 3100 FORMAT(I2)
      IF(TPDRR.GE.3) STOP'NON-EXISTANT READ DRIVE'
      GOTO 370
C
C  CHECK BOT FLAG
C
```

```
 380  IEOTR=0
      IF((ISTUS.AND."40).LE.0) GOTO 400
      CALL ISPFNW("376,ICHN)
      CALL ISPFNW("376,ICHN)
C
C  OPEN DISC FILE FOR STORAGE OF DATA
C
 400  CONTINUE
      IF(IENTER(NCHN,FSPECR,IBKTOT).LE.0) STOP'DISC FILE ENTER ERR'
C
C  LOOP OVER ALL GATHERS AND TRACES TO BE TRANSFERED
C
      DO 800 I=1,IGATHR
      DO 700 J=1,ITRCIN
      ITRI=J
 500  CALL TRREAD(SEIS,ICDPNO,IWORDR,ITRI,0,ISTOP)
      IF(ISTOP.NE.0) STOP'TRACE READ ERROR'
      IF(I.GT.1.OR.J.GT.1) GOTO 650
C
C  CHECK FIRST GATHER NUMBER
C
      IF(ISHOT.GT.0) GOTO 550
      CALL JJCVT(JCDPNO)
      CALL IJCVT(JCDPNO,ICDPNO)
      CALL JJCVT(JCDPNO)
      IF(ICDPNO.LT.ISTART) GOTO 520
      CALL JJCVT(JCDPTR)
      CALL IJCVT(JCDPTR,ICDPTR)
      CALL JJCVT(JCDPTR)
      IF(ICDPTR.LE.1) GOTO 600
 520  CALL ISPFNW("376,ICHN)
      GOTO 500
C
C  COME HERE IF SHOT DATA
C
 550  CALL JJCVT(JFNUM)
      CALL IJCVT(JFNUM,IFNUM)
      CALL JJCVT(JFNUM)
      IF(IFNUM.LT.ISTART) GOTO 570
      CALL JJCVT(JTRAS)
      CALL IJCVT(JTRAS,ITRAS)
      CALL JJCVT(JTRAS)
      IF(ITRAS.LE.1) GOTO 600
 570  CALL ISPFNW("376,ICHN)
      GOTO 500
C
C  WRITE OUT FILE HEADER
C
 600  IF(NDELAY.LT.0) NDELAY=IDELAY
      DO 610 K=1,60
 610  TRHDBF(K)=SEIS(K)
      HBLK(18)=ITE-ITS+1
      HBLK(55)=NDELAY
      HBLK(58)=NSMPOT
      HBLK(15)=1
      IF(IWRITW(256,HBLK,0,NCHN).NE.256) STOP'FILE HEADER WRITE ERR'
C
C  CHECK DELAY ON TRACE AND MOVE IF NECESSARY, REMOVE UNWANTED SAMPLES
C
 650  IF(J.LT.ITS.OR.J.GT.ITE) GOTO 700
      NSOFF=0
      IAPST=IFIX(FLOAT(IDELAY-NDELAY)*FSAMP)
      IF(IAPST.GE.0) GOTO 720
      NSOFF=-IAPST
      IAPST=0
 720  CALL VCLR(0,1,4096)
      CALL APWAIT
      IF(IDEAD.EQ.0.AND.ITYPE.GT.1) GOTO 730
      CALL APPTXM(SEIS(61+NSOFF),IAPST,(NSMPIN-NSOFF),2)
 730  CALL APGTXM(SEIS(61),0,NSMPOT,2)
      CALL APWAIT
      IF(IWDRXM(SEIS(61),NCHN,IWORDW,IBLK,0).NE.IWORDW) STOP
```

```
     +'TRACE WRITE ERROR'
      IBLK=IBLK+NBLK
 700  CONTINUE
 800  CONTINUE
C
C  RETURN
C
      NSMPIN=NSMPOT
      ITRCIN=ITE-ITS+1
      ITS=1
      ITE=ITRCIN
      CALL CLOSEC(ICHN)
      CALL CLOSEC(NCHN)
      CALL IFREEC(NCHN)
      RETURN
      END




      SUBROUTINE BACKGD(FSPECR)
C
C  THIS ROUTINE CREATES THE BACKGOUND TO GO WITH SECTION AND
C  GATHER PLOTS
C
C  WRITTEN R.W.HOBBS 1985
C
      VIRTUAL VEL(2,11,100),ATIME(10),AVRMS(10),AVINT(10)
      REAL*8 FSPECR
      INTEGER*4 JCDPNO,JFNUM
      INTEGER*2 PARM(50),HBLK(256)
      LOGICAL*1 OUTBUF(240),TITLE(20),ANNO(80)
C
      COMMON/PASS/ PARM
      COMMON/DATBUF/ OUTBUF
      COMMON /PPEP1/ IX1,IY1,IX2,IY2,ISCAN,NSCAN,NBAND,NPIS,NIP0,
     +NIPM1,LYNES,NIBSX,MSGLVL,XDOTS,YDOTS,PREF(2),
     +RORG(2),PORT(2,2),INND(4),ALMT,FACT,JPEN,XOFF,
     +XFAC,YOFF,YFAC,NBITS,NBITM1,NBYTES,NBYTM1,MSK,LMSK
      COMMON/CNTRL/ NHSUM,IBACK,IVEL,IDOC,NANNO,IDEAD
C
      EQUIVALENCE (PARM(5),IPLOT),(PARM(6),ISHOT),
     +(PARM(7),ISTART),(PARM(8),IGATHR),(PARM(9),ITRCIN),
     +(PARM(10),NSMPIN),(PARM(11),NDELAY),(PARM(12),NLENG),
     +(PARM(13),TLENG),(PARM(15),FSAMP),(PARM(17),IAGC),
     +(PARM(18),NIBS),(PARM(19),GAIN),(PARM(21),ICLIP),
     +(PARM(22),NCLP),(PARM(23),IPOL),(PARM(24),IREV),
     +(PARM(25),ISDLY),(PARM(26),ISAMPS),(PARM(27),ICHAN),
     +(PARM(28),LINE),(PARM(29),TITLE(1)),(PARM(43),ITS),
     +(PARM(44),ITE)
      EQUIVALENCE (HBLK(5),JFNUM),(HBLK(11),JCDPNO)
C
      DATA LMASK/"177777/
C
C  LOOKUP DISC FILE AND READ FIRST BLOCK
C
      NCHN=IGETC()
      IF(LOOKUP(NCHN,FSPECR).LT.0) STOP'DISC FILE LOOKUP ERR'
      IF(IREADW(256,HBLK,0,NCHN).NE.256) STOP'FILE HEADER READ ERR'
      CALL CLOSEC(NCHN)
      CALL IFREEC(NCHN)
      CALL JJCVT(JCDPNO)
      CALL IJCVT(JCDPNO,ICDPNO)
      ISTART=ICDPNO
      IF(ISHOT.EQ.0) GOTO 50
      CALL JJCVT(JFNUM)
      CALL IJCVT(JFNUM,IFNUM)
      ISTART=IFNUM
 50   IF(ISTART.EQ.0) ISTART=1
      ITYPE=HBLK(15)
      IF(NDELAY.LT.0) NDELAY=HBLK(55)
```

```
      NTRCIN=ITE-ITS+1
C
C  SET UP DEFAULT VALUES
C
      D9=90.0
      X9=999.0
      TPAPER=10.56
      TUPPER=10.55
      TSEC=0.001*FLOAT(NLENG)
      TINDEL=0.001*FLOAT(NDELAY)
      TSPACE=FLOAT(NIBS)/(200.0*FLOAT(NHSUM))
      IEND=ISTART+(IGATHR*NHSUM)-1
      CHIGH=FLOAT(2*NIBS)/200.0
      CLOW=-CHIGH
      IF(ICLIP.EQ.1) CLOW=FLOAT(NCLP)/200.0
      IF(IPOL.GE.0) GOTO 100
      CSAW=CHIGH
      CHIGH=-CLOW
      CLOW=-CSAW
  100 TINT=1.0
      TSCALE=TLENG/TSEC
      TOTAL=10.56
      IF(IBACK.EQ.0) GOTO 220
      TWIDTH=(FLOAT(IEND-ISTART+1)*FLOAT(NTRCIN)
     $        +(5.0*FLOAT(NHSUM)))*TSPACE
      IF(IPLOT.EQ.0) TWIDTH=TWIDTH+(FLOAT(IEND-ISTART)*5.0*TSPACE)
      TOTAL=TWIDTH+6.0
      IF(IVEL.EQ.1) TOTAL=TOTAL+24.0
      IF(IDOC.EQ.1) TOTAL=TOTAL+10.0
C
C  INITIALIZE PLOT
C
  220 PORT(1,1)=TOTAL
      CALL PLOT(2.5,0.0,-3)
C
C  ANNOTATE
C
      CALL NEWPEN(1)
      PSIZE=0.1
      X=-2.0
      Y=1.0
      CALL SYMBOL(X,Y,PSIZE,TITLE,D9,20)
      CALL SYMBOL(X9,X9,PSIZE,'      LINE : ',D9,12)
      FNUM=LINE
      CALL NUMBER(X9,X9,PSIZE,FNUM,D9,-1)
      CALL SYMBOL(X9,X9,PSIZE,'      START GATHER NO. ',D9,22)
      FNUM=ISTART
      CALL NUMBER(X9,X9,PSIZE,FNUM,D9,-1)
      CALL SYMBOL(X9,X9,PSIZE,'      END GATHER NO. ',D9,20)
      FNUM=IEND
      CALL NUMBER(X9,X9,PSIZE,FNUM,D9,-1)
      X=-1.75
      IF(ISHOT.EQ.0)CALL SYMBOL(X,Y,PSIZE,'COMMON MID-POINT',D9,16)
      IF(ISHOT.EQ.1)CALL SYMBOL(X,Y,PSIZE,'COMMON SHOT POINT',D9,17)
      IF(ITYPE.NE.9) CALL SYMBOL(X9,X9,PSIZE,'      ',D9,5)
      IF(ITYPE.EQ.9)
     +CALL SYMBOL(X9,X9,PSIZE,'      AUTOCORRELATION ',D9,21)
      IF(IPLOT.EQ.0)CALL SYMBOL(X9,X9,PSIZE,'GATHER PLOT',D9,11)
      IF(IPLOT.EQ.1)CALL SYMBOL(X9,X9,PSIZE,'SECTION PLOT',D9,12)
      CALL SYMBOL(X9,X9,PSIZE,'      ',D9,5)
      IF(IPOL.GT.0)CALL SYMBOL(X9,X9,PSIZE,'POSITIVE LOBE SHADED',D9,20)
      IF(IPOL.LT.0)CALL SYMBOL(X9,X9,PSIZE,'NEGATIVE LOBE SHADED',D9,20)
      IF(IPOL.EQ.0) CALL SYMBOL(X9,X9,PSIZE,'WIGGLY LINE PLOT',D9,16)
      X=-1.50
      CALL SYMBOL(X,Y,PSIZE,'TRACE SPACING ',D9,14)
      FNUM=TSPACE*FLOAT(NHSUM)
      CALL NUMBER(X9,X9,PSIZE,FNUM,D9,3)
      CALL SYMBOL(X9,X9,PSIZE,' INCHES',D9,7)
      CALL SYMBOL(X9,X9,PSIZE,'      TRACE LENGTH ',D9,18)
      CALL NUMBER(X9,X9,PSIZE,TLENG,D9,2)
      CALL SYMBOL(X9,X9,PSIZE,' INCHES',D9,7)
      IF(IAGC.EQ.1)
```

```
     +CALL SYMBOL(X9,X9,PSIZE,'     50 SAMPLE AGC APPLIED',D9,26)
      X=-1.25
      CALL SYMBOL(X,Y,PSIZE,'PLOTTING GAIN ',D9,14)
      ENCODE(10,2300,AGAIN,ERR=225) GAIN
 2300 FORMAT(1PE10.3)
      CALL SYMBOL(X9,X9,PSIZE,AGAIN,D9,10)
 225  CALL SYMBOL(X9,X9,PSIZE,'     LOW CLIP LEVEL ',D9,20)
      CALL NUMBER(X9,X9,PSIZE,CLOW,D9,3)
      CALL SYMBOL(X9,X9,PSIZE,' INCHES',D9,7)
      CALL SYMBOL(X9,X9,PSIZE,'     HIGH CLIP LEVEL ',D9,21)
      CALL NUMBER(X9,X9,PSIZE,CHIGH,D9,3)
      CALL SYMBOL(X9,X9,PSIZE,' INCHES',D9,7)
C
C  PLOT MINIMUM TIME AXIS
C
      CALL NEWPEN(3)
      CALL PLOT(-0.3,0.0,3)
      CALL PLOT(-0.3,TUPPER,2)
      CALL NEWPEN(1)
      CALL PLOT(0.0,0.0,3)
      CALL PLOT(0.0,TUPPER,2)
C
C  ANNOTATE MARGIN
C
      Y=10.3
      X=-0.1
      PSIZE=0.08
      CALL SYMBOL(X,Y,PSIZE,'SEC',90.0,3)
      IF(TSCALE.LT.5.0) GOTO 260
      TINT=0.1
 260  X=-0.25
      MINT=IFIX((TINDEL+TINT)/TINT)
      TMIN=FLOAT(MINT)*TINT
      TBASE=(TMIN-TINDEL)*TSCALE
      TPLOT=10.2-TBASE
      FNUM=TMIN
 270  IF(FNUM.LT.10.0) CALL NUMBER(X,TPLOT,PSIZE,FNUM,0.0,1)
      IF(FNUM.GE.10.0) CALL NUMBER(X,TPLOT,PSIZE,FNUM,0.0,0)
      TPLOT=TPLOT-(TSCALE*TINT)
      IF(TPLOT.LE.0.0) GOTO 300
      FNUM=FNUM+TINT
      GOTO 270
 300  ISDLY=IFIX(((TMIN-TINDEL)*TSCALE*200.0)+0.5)
      ISAMPS=IFIX((TINT*TSCALE*200.0)+0.5)
      IF(IBACK.EQ.1) GOTO 400
      GOTO 900
C
C  MAX ANNOTATION
C
 400  TYPE 2000
 2000 FORMAT(' PLOT BACKGROUND',/)
      CALL NEWPEN(3)
      CALL PLOT(-0.3,0.0,3)
      CALL PLOT(TWIDTH+0.3,0.0,2)
      CALL PLOT(TWIDTH+0.3,TUPPER,2)
      CALL PLOT(-0.3,TUPPER,2)
      CALL NEWPEN(1)
      CALL PLOT(-0.3,10.24,3)
      CALL PLOT(TWIDTH+0.3,10.24,2)
      CALL PLOT(TWIDTH,0.0,3)
      CALL PLOT(TWIDTH,TUPPER,2)
C
C  ANNOTATE RIGHT MARGIN
C
      Y=10.3
      X=TWIDTH+0.2
      CALL SYMBOL(X,Y,PSIZE,'SEC',90.0,3)
      X=TWIDTH+0.05
      TPLOT=10.2-TBASE
      FNUM=TMIN
 410  IF(FNUM.LT.10.0) CALL NUMBER(X,TPLOT,PSIZE,FNUM,0.0,1)
      IF(FNUM.GE.10.0) CALL NUMBER(X,TPLOT,PSIZE,FNUM,0.0,0)
```

```
      TPLOT=TPLOT-(TSCALE*TINT)
      IF(TPLOT.LE.0.0) GOTO 420
      FNUM=FNUM+TINT
      GOTO 410
C
C  ANNOTATE TOP MARGIN
C
 420  CONTINUE
      IF(IPLOT.EQ.1) GOTO 450
      GBEG=3.0*TSPACE
      GINT=1.0
      GSTART=ISTART
      GSPACE=FLOAT(NTRCIN+5)*TSPACE
 430  IF(GSPACE.GT.0.56) GOTO 500
      GINT=2.0*GINT
      GSPACE=2.0*GSPACE
      GOTO 430
 450  GINT=NANNO
      IF(GINT.EQ.0.0) GINT=10.0
 460  GSPACE=GINT*TSPACE
      IF(GSPACE.GE.0.56) GOTO 470
      GINT=2.0*GINT
      GOTO 460
 470  IF(IREV.EQ.1) GOTO 490
      GSTART=GINT*FLOAT(IFIX(FLOAT(ISTART)/GINT)+1)
 480  GBEG=(GSTART-FLOAT(ISTART)+3.0*FLOAT(NHSUM))*TSPACE
      IF(GBEG.GT.0.28) GOTO 500
      GSTART=GSTART+GINT
      GOTO 480
 490  GSTART=GINT*FLOAT(IFIX(FLOAT(IEND)/GINT))
      GINT=-GINT
 495  GBEG=(FLOAT(IEND)-GSTART+3.0*FLOAT(NHSUM))*TSPACE
      IF(GBEG.GT.0.28) GOTO 500
      GSTART=GSTART+GINT
      GOTO 495
 500  X=GBEG
      Y=10.36
      FNUM=GSTART
 510  XP=X
      IF(IPLOT.EQ.0) GOTO 520
      LMSK="401
      MSK=-1
      CALL PLOT(X,0.0,3)
      CALL PLOT(X,10.24,2)
      LMSK=-1
      MSK=0
      IF(FNUM.GT.9999.0) XP=XP-0.04
      IF(FNUM.GT.999.0) XP=XP-0.04
      IF(FNUM.GT.99.0) XP=XP-0.04
      IF(FNUM.GT.9.0) XP=XP-0.04
      XP=XP-0.02
 520  CALL NUMBER(XP,Y,PSIZE,FNUM,0.0,-1)
      X=X+GSPACE
      FNUM=FNUM+GINT
      IF(X.GT.TWIDTH-0.28) GOTO 600
      GOTO 510
C
C  PLOT VELOCITY GATHERS AS REQUIRED
C
 600  IF(IVEL.EQ.0) GOTO 700
      TYPE 2100
 2100 FORMAT(' PLOT VELOCITY DATA',/)
C
C  READ VELOCTIY DATA
C
      IVCNT=0
      OPEN(UNIT=1,NAME='DK6:VELRMS.DAT',TYPE='OLD',ERR=690)
 610  READ(1,3000,END=620) NCDP,NLYR
 3000 FORMAT(2I6)
      IVCNT=IVCNT+1
      VEL(1,1,IVCNT)=FLOAT(NCDP)
      VEL(2,1,IVCNT)=FLOAT(NLYR)
```

```
      DO 615 J=1,NLYR
      READ(1,3100) VEL(1,J+1,IVCNT),VEL(2,J+1,IVCNT)
 3100 FORMAT(2F10.0)
 615  CONTINUE
      GOTO 610
 620  CONTINUE
      CLOSE(UNIT=1)
C
C  MARK STARS AT VELOCITY ANALYSIS POINTS
C
      Y=10.25
      IBEG=ISTART
      IF(IREV.EQ.1) IBEG=IEND
      DO 630 I=1,IVCNT
      IF(IFIX(VEL(1,1,I)).LT.ISTART) GOTO 630
      IF(IFIX(VEL(1,1,I)).GT.IEND) GOTO 630
      ISHIFT=IABS(IFIX(VEL(1,1,I))-IBEG)+(3*NHSUM)
      X=FLOAT(ISHIFT)*TSPACE-0.04
      CALL SYMBOL(X,Y,PSIZE,14,0.0,-1)
 630  CONTINUE
C
C  PLOT VELOCITY BOXES
C  RE-ORIGIN
C
      CALL PLOT(TWIDTH+1.6,0.0,-3)
      AMARK=0.0
      VPLT1=0.0
      DO 640 I=1,IVCNT
      IF(IFIX(VEL(1,1,I)).LT.ISTART) GOTO 640
      IF(IFIX(VEL(1,1,I)).GT.IEND) GOTO 640
      NLYR=IFIX(VEL(2,1,I))
      DO 650 J=1,NLYR
      ATIME(J)=0.001*VEL(1,J+1,I)
      AVRMS(J)=VEL(2,J+1,I)
      IF(J.EQ.1) AVINT(J)=AVRMS(J)
      IF(J.EQ.1) GOTO 650
      VINTSQ=((AVRMS(J)*AVRMS(J)*ATIME(J))-
     $(AVRMS(J-1)*AVRMS(J-1)*ATIME(J-1)))/(ATIME(J)-ATIME(J-1))
      IF(VINTSQ.LT.0.0) VINTSQ=0.0
      AVINT(J)=SQRT(VINTSQ)
 650  CONTINUE
C
C  SET UP VELOCITY BOX SIZES
C
      VBSIZE=0.27
      VBLINE=0.11
      VBSTX=VPLT1-0.5*VBSIZE*3.0
      VBSTY=10.24*(1.0-AMARK*0.25)-13.0*VBLINE
C
C  PLOT BOXES
C
      CALL PLOT(VBSTX+(3.0*VBSIZE),VBSTY-VBLINE,3)
      CALL PLOT(VBSTX+(3.0*VBSIZE),VBSTY,2)
      CALL PLOT(VBSTX,VBSTY,2)
      CALL PLOT(VBSTX,VBSTY-VBLINE,2)
      CALL GRID(VBSTX,VBSTY-VBLINE,3,VBSIZE,11,VBLINE,LMASK)
      PSIZE=0.05
      X=VPLT1-2.0*PSIZE
      Y=10.24*(1.0-AMARK*0.25)-2.0*VBLINE+0.025
      CALL NUMBER(X,Y,PSIZE,VEL(1,1,I),0.0,-1)
      Y=Y-VBLINE
      X=VBSTX+0.1*VBSIZE
      XP=X
      CALL SYMBOL(X,Y,PSIZE,'TIME',0.0,4)
      X=X+VBSIZE
      CALL SYMBOL(X,Y,PSIZE,'VRMS',0.0,4)
      X=X+VBSIZE
      CALL SYMBOL(X,Y,PSIZE,'VINT',0.0,4)
      Y=Y-VBLINE
      X=XP
C
C  PUT IN THE NUMBERS
```

```
C
      DO 660 J=1,NLYR
      IF(ATIME(J).LT.10.0) CALL NUMBER(X,Y,PSIZE,ATIME(J),0.0,3)
      IF(ATIME(J).GE.10.0) CALL NUMBER(X,Y,PSIZE,ATIME(J),0.0,2)
      X=X+VBSIZE
      CALL NUMBER(X,Y,PSIZE,AVRMS(J),0.0,3)
      X=X+VBSIZE
      CALL NUMBER(X,Y,PSIZE,AVINT(J),0.0,3)
      X=XP
 660  Y=Y-VBLINE
C
C  MOVE TO NEXT BOX LOCATION
C
      IF(AMARK.NE.3.0) GOTO 670
      AMARK=0.0
      VPLT1=VPLT1+1.0
      GOTO 640
 670  AMARK=AMARK+1.0
 640  CONTINUE
      IF(AMARK.EQ.0.0) VPLT1=VPLT1-1.0
      GOTO 700
 690  TYPE 6900
 6900 FORMAT(/,' NO VELOCITY DATA FILE DK6:VELRMS.DAT',/)
C
C  DOCUMENTATION SECTION
C  RE-ORIGIN
C
 700  IF(IDOC.EQ.0) GOTO 900
      TYPE 2200
 2200 FORMAT(' PLOT DOCUMENTATION',/)
      X=TWIDTH+1.6
      IF(IVEL.NE.0) X=VPLT1+1.6
      CALL PLOT(X,0.0,-3)
C
C  PLOT BOX
C
      CALL NEWPEN(4)
      CALL PLOT(0.0,0.0,3)
      CALL PLOT(0.0,10.24,2)
      CALL PLOT(8.5,10.24,2)
      CALL PLOT(8.5,0.0,2)
      CALL PLOT(0.0,0.0,2)
      CALL NEWPEN(1)
      CALL PLOT(0.05,0.05,3)
      CALL PLOT(0.05,10.19,2)
      CALL PLOT(8.45,10.19,2)
      CALL PLOT(8.45,0.05,2)
      CALL PLOT(0.05,0.05,2)
C
C  PUT IN TITLE
C
      CALL NEWPEN(3)
      PSIZE=0.4
      X=0.9
      Y=9.64
      CALL SYMBOL(X,Y,PSIZE,'DURHAM UNIVERSITY',0.0,17)
      CALL NEWPEN(1)
      X=1.6
      Y=9.2
      PSIZE=0.3
      CALL SYMBOL(X,Y,PSIZE,'SEISMIC PROCESSING',0.0,18)
      PSIZE=0.2
      X=1.0
      Y=8.9
      CALL SYMBOL(X,Y,PSIZE,TITLE,0.0,20)
      X=5.75
      CALL SYMBOL(X,Y,PSIZE,'LINE : ',0.0,7)
      FNUM=LINE
      CALL NUMBER(999.0,999.0,PSIZE,FNUM,0.0,-1)
      CALL PLOT(0.0,8.8,3)
      CALL PLOT(8.5,8.8,2)
      X=0.25
```

```
      Y=8.5
      CALL SYMBOL(X,Y,PSIZE,'SYSTEM PARAMETERS',0.0,17)
C
C  OPEN DOCUMENT DATA FILE DK6:DOC.DAT
C
      OPEN(UNIT=1,NAME='DK6:DOC.DAT',TYPE='OLD',ERR=790)
 7000 FORMAT(80A1)
      PSIZE=0.1
      Y=8.0
C
C  READ DATA FILE AND PLOT
C
 750  READ(1,7000,END=800)(ANNO(I),I=1,80)
      CALL SYMBOL(X,Y,PSIZE,ANNO,0.0,80)
      Y=Y-0.15
      IF(Y.LE.0.1) GOTO 800
      GOTO 750
 800  CLOSE(UNIT=1)
      GOTO 900
 790  TYPE 7900
 7900 FORMAT(/,' NO DOCUMENT FILE DK6:DOC.DAT',/)
C
C  RETURN
C
 900  CONTINUE
      RETURN
      END
```

```
C
C  THIS IS A FOURIER ANALYSIS ROUTINE
C  READ DATA FROM A DISC FILE AND OUTPUTS THREE SPECTRA
C  POWER, AMPLITUDE AND PHASE
C
C  WRITTEN BY R.W.HOBBS 1985, FROM ORIGINAL BY M.J.POULTER 1982
C
      VIRTUAL SEIS(4156),ARRAY(6150)
      REAL*8 FSPECR,FSPECW
      REAL*4 PLTRSM(2),ALIM(4),CONST(3)
      INTEGER*2 PARM(50),HBLK(256)
      LOGICAL*1 TITLE(20),AGAIN(20)
C
      COMMON/PASS/ PARM
      COMMON /PPEP1/ IX1,IY1,IX2,IY2,ISCAN,NSCAN,NBAND,NPIS,NIP0,
     +NIPM1,LYNES,NIBSX,MSGLVL,XDOTS,YDOTS,PREF(2),
     +RORG(2),PORT(2,2),INND(4),ALMT,FACT,JPEN,XOFF,
     +XFAC,YOFF,YFAC,NBITS,NBITM1,NBYTES,NBYTM1,MSK,LMSK
C
      EQUIVALENCE (PARM(5),IPLOT),(PARM(6),ISHOT),
     +(PARM(7),ISTART),(PARM(8),IGATHR),(PARM(9),ITRCIN),
     +(PARM(10),NSMPIN),(PARM(11),NDELAY),(PARM(12),NLENG),
     +(PARM(13),TLENG),(PARM(15),FSAMP),(PARM(17),IAGC),
     +(PARM(18),NIBS),(PARM(19),GAIN),(PARM(21),ICLIP),
     +(PARM(22),NCLP),(PARM(23),IPOL),(PARM(24),IREV),
     +(PARM(25),ISDLY),(PARM(26),ISAMPS),(PARM(27),ICHAN),
     +(PARM(28),LINE),(PARM(29),TITLE(1)),(PARM(39),FSPECW)
C
      DATA APHN/3RAP /,DRHN/3RDR /
      DATA MASK/"104210/
      DATA PLTRSM(1)/6RDK1PLT/,PLTRSM(2)/6RRSMSAV/
C
      CALL RCHAIN(ICHFLG,PARM,50)
      IF(ICHFLG.NE.-1) STOP'THIS PROGRAM MUST BE CALLED FROM PLOT'
C
C  FETCH HANDLERS
C
      IF(IFETCH(DRHN).NE.0) STOP'DR FETCH ERROR'
      IF(IFETCH(APHN).NE.0) STOP'AP FETCH ERROR'
      CALL APINIT(0,0,IAP)
C
C  SET UP CONSTANTS
C
      FSPECR=FSPECW
      TSAMP=0.001/FSAMP
      NSAMP=FLOAT(NLENG)*FSAMP
      NSAMP2=2
 10   IF(NSAMP2.GE.NSAMP) GOTO 20
      NSAMP2=2*NSAMP2
      GOTO 10
 20   NCMPLX=(NSAMP2/2)+1
      NBLK=NSMPIN/128
      IF(MOD(NSMPIN,128).NE.0) NBLK=NBLK+1
      IWORDR=2*NSMPIN
      X9=999.0
      D9=90.0
      PSIZE=0.1
      YSIZE=9.0
      XSIZE=0.75*YSIZE
      YOFST=1.0
      XOFST=1.0
      XPAGE=8.5-XOFST-XSIZE
      DELF=1.0/(FLOAT(NSAMP2)*TSAMP)
      FLIM=0.5*DELF*FLOAT(NSAMP2)
      FBASE=0.1
      IBASE=-1
 60   IF(FLIM.LE.FBASE) GOTO 70
      FBASE=FBASE*10.0
      IBASE=IBASE+1
      GOTO 60
 70   FBASE=FBASE*0.1
      IBASE=IBASE-1
```

```
      FMAX=FLOAT(IFIX(FLIM/FBASE)+1)
      FEND=FMAX*FBASE
      DFPLOT=YSIZE*DELF/FEND
      CONST(1)=-XSIZE
      CONST(2)=180.0/3.14159
      CONST(3)=3.14159
C
C  LOOKUP DISC FILE
C
      ICHN=IGETC()
      IF(LOOKUP(ICHN,FSPECR).LT.0) STOP'FANL LOOKUP ERR'
C
C  READ HEADER BLOCK
C
      IF(IREADW(256,HBLK,0,ICHN).NE.256) STOP'FANL H/BLK READ ERR'
      IDELAY=HBLK(55)
      IF(NDELAY.LT.0) NDELAY=IDELAY
      NSOFF=0
      IAPST=IFIX(FLOAT(IDELAY-NDELAY)*FSAMP)
      IF(IAPST.GE.0) GOTO 50
      NSOFF=-IAPST
      IAPST=0
 50   NSMPUT=MIN0(IFIX(FLOAT(IDELAY)*FSAMP)+NSMPIN,
     +IFIX(FLOAT(NDELAY+NLENG)*FSAMP))-
     +IFIX(FLOAT(MAX0(IDELAY,NDELAY))*FSAMP)
      IAPST=IAPST+12288
      IAPND=IAPST+NSMPUT
C
C  PUT CONSTANTS IN AP AT ADDESS 21000
C
      CALL APWD
      CALL APPUT(CONST,21000,3,2)
      CALL APWD
C
C  PROCESS REQUIRED CHANNEL
C
      IF(ICHAN.EQ.0) GOTO 900
      NBLKST=(ICHAN-1)*NBLK+1
      IF(IRDRXM(SEIS(1),ICHN,IWORDR,NBLKST,0).NE.IWORDR) STOP
     +'FANL TRACE READ ERR'
      CALL APWAIT
      CALL VCLR(0,1,16386)
      CALL APWAIT
      CALL APPTXM(SEIS(1+NSOFF),IAPST,NSMPUT,2)
      CALL APWAIT
C
C  TAPER BOTH ENDS OF DATA TO AVOID WRAP ROUND DISCONTINUITY
C
      CALL TAPER(IAPST,1,IAPST,1,10)
      CALL TAPER(IAPND-1,-1,IAPND-1,-1,10)
C
C  FOURIER TRANSFORM
C
      CALL RFFT(12288,NSAMP2,1)
      CALL RFFTSC(12288,NSAMP2,3,1)
C
C  FORM POWER SPECTRUM AS SQUARE OF MAGNITUDE OF AMPLITUDE SPECTRUM
C
      CALL CVMUL(12288,2,12288,2,0,2,NCMPLX,-1)
C
C  FORM THREE VECTORS 0=POWER, 4096=AMPLITUDE, 8192=PHASE
C
      CALL POLAR(12288,2,12288,2,NCMPLX)
      CALL VMOV(0,2,0,1,NCMPLX)
      CALL VMOV(12288,2,4096,1,NCMPLX)
      CALL VMOV(12289,2,12288,1,NCMPLX)
C
C  UNWRAP PHASE, NORMALIZE BOTH POWER AND AMPLITUDE, SAVE PLOT LIMITS
C
      CALL SHPHU(12288,1,21002,8192,1,NCMPLX)
      CALL VSDIV(8192,1,21002,8192,1,NCMPLX)
      CALL MAXV(0,1,20000,NCMPLX)
```

```
      CALL MAXV(4096,1,20001,NCMPLX)
      CALL MINV(8192,1,20002,NCMPLX)
      CALL MAXV(8192,1,20003,NCMPLX)
      CALL VSDIV(0,1,20000,0,1,NCMPLX)
      CALL VSDIV(4096,1,20001,4096,1,NCMPLX)
      CALL VSMUL(0,1,21000,0,1,NCMPLX)
      CALL VSMUL(4096,1,21000,4096,1,NCMPLX)
C
C  SORT OUT LIMITS ON UNWRAPPED PHASE PLOT
C
      CALL APWR
      CALL APGET(ALIM,20000,4,2)
      CALL APWD
      XL=ALIM(4)-ALIM(3)
      XFAC=1.0
 200  XS=1.0*XFAC
      IF(XL.LE.20.0*XFAC) GOTO 300
      XS=2.0*XFAC
      IF(XL.LE.40.0*XFAC) GOTO 300
      XS=5.0*XFAC
      IF(XL.LE.100.0*XFAC) GOTO 300
      XFAC=10.0*XFAC
      GOTO 200
 300  XMAX=SIGN(XS*FLOAT(IFIX(ABS(ALIM(4))/XS)+1),ALIM(4))
      XMIN=SIGN(XS*FLOAT(IFIX(ABS(ALIM(3))/XS)+1),ALIM(3))
      XL=XMAX-XMIN
      CALL APWD
      CALL APPUT(XL,21003,1,2)
      CALL APPUT(XMIN,21004,1,2)
      CALL APWD
      CALL VSUB(21004,0,8192,1,8192,1,NCMPLX)
      CALL VSDIV(8192,1,21003,8192,1,NCMPLX)
      CALL VSMUL(8192,1,21000,8192,1,NCMPLX)
C
C  GET DATA OUT OF AP AND RELEASE AP, CANNOT USE PLOT ROUTINES OTHERWISE
C
      CALL APWD
      CALL APGTXM(ARRAY(1),0,NCMPLX,2)
      CALL APGTXM(ARRAY(2051),4096,NCMPLX,2)
      CALL APGTXM(ARRAY(4101),8192,NCMPLX,2)
      CALL APWD
      CALL APRLSE
C
C  CREATE PLOT FILE
C
      DO 700 J=1,3
      IF(J.EQ.1) CALL PLOTS(0,0,0)
      IF(J.EQ.1) PORT(1,1)=30.0
      IF(J.NE.1) CALL PLOT(XPAGE,-YOFST,-3)
      CALL NEWPEN(1)
      DELX=XSIZE/10.0
      NX=10
      XINC=0.1
      NY=IFIX(FLOAT(10.0*FMAX))
      DELY=YSIZE/FLOAT(NY)
      IF(J.LE.2) GOTO 500
      NX=IFIX(XL/XS)
      IF(NX.EQ.0) NX=1
      DELX=XSIZE/FLOAT(NX)
 500  CALL PLOT(XOFST,YOFST,-3)
      CALL GRID(0.,0.,NX,DELX,NY,DELY,MASK)
C
C  ANNOTATE AXIS
C
      CALL PLOT(XSIZE,0.,-3)
      NX=NX+1
      X=0.0
      Y=-0.3
      XNUM=0.0
      IF(J.LE.2) GOTO 505
      XNUM=XMIN
      XINC=XS
```

```
505   DO 510 K=1,NX
      CALL PLOT(X,0.,3)
      CALL PLOT(X,YSIZE,2)
      IF(J.EQ.3) GOTO 507
      CALL NUMBER(X+0.05,Y,PSIZE,XNUM,D9,1)
      GOTO 508
507   Y=-0.1
      IF(ABS(XNUM).GE.10.0) Y=Y-0.1
      IF(ABS(XNUM).GE.100.0) Y=Y-0.1
      IF(ABS(XNUM).GE.1000.0) Y=Y-0.1
      IF(ABS(XNUM).GE.10000.0) Y=Y-0.1
      IF(ABS(XNUM).GE.100000.0) Y=Y-0.1
      IF(XNUM.LT.0) Y=Y-0.1
      CALL NUMBER(X+0.05,Y,PSIZE,XNUM,D9,-1)
508   XNUM=XNUM+XINC
510   X=X-DELX
      NY=IFIX(FLOAT(NY)*0.1)
      DELY=YSIZE/FLOAT(NY)
      NY=NY+1
      YNUM=0.0
      Y=0.0
      YINC=1.0
      X=0.15
      DO 520 K=1,NY
      CALL PLOT(0.,Y,3)
      CALL PLOT(-XSIZE,Y,2)
      CALL NUMBER(X,Y-0.03,PSIZE,YNUM,D9,-1)
      YNUM=YNUM+YINC
520   Y=Y+DELY
C
C  ANNOTATE PLOT
C
      CALL SYMBOL(-7.2,0.,PSIZE,TITLE,D9,20)
      CALL SYMBOL(X9,X9,PSIZE,'    LINE : ',D9,11)
      FNUM=FLOAT(LINE)
      CALL NUMBER(X9,X9,PSIZE,FNUM,D9,-1)
      CALL SYMBOL(X9,X9,PSIZE,'    GATHER : ',D9,13)
      FNUM=FLOAT(ISTART)
      CALL NUMBER(X9,X9,PSIZE,FNUM,D9,-1)
      CALL SYMBOL(X9,X9,PSIZE,'    TRACE : ',D9,12)
      FNUM=FLOAT(ICHAN)
      CALL NUMBER(X9,X9,PSIZE,FNUM,D9,-1)
      IF(J.EQ.1) CALL SYMBOL(X9,X9,PSIZE,'    POWER SPECTRUM',D9,18)
      IF(J.EQ.2) CALL SYMBOL(X9,X9,PSIZE,'    AMPLITUDE SPECTRUM',D9,22)
      IF(J.EQ.3) CALL SYMBOL(X9,X9,PSIZE,'    PHASE SPECTRUM',D9,18)
      CALL SYMBOL(-7.0,0.,PSIZE,'START TIME FOR ANALYSIS : ',D9,26)
      FNUM=FLOAT(NDELAY)
      CALL NUMBER(X9,X9,PSIZE,FNUM,D9,-1)
      CALL SYMBOL(X9,X9,PSIZE,
     +' MILLISECS    LENGTH OF ANALYSIS : ',D9,35)
      FNUM=FLOAT(NLENG)
      CALL NUMBER(X9,X9,PSIZE,FNUM,D9,-1)
      CALL SYMBOL(X9,X9,PSIZE,' MILLISECS',D9,10)
      IF(J.EQ.3) GOTO 530
      CALL SYMBOL(0.3,0.,PSIZE,'VERTICAL SCALING FACTOR : ',D9,26)
      IF(J.EQ.1) ENCODE(10,5000,AGAIN,ERR=540) ALIM(1)
      IF(J.EQ.2) ENCODE(10,5000,AGAIN,ERR=540) ALIM(2)
5000  FORMAT(1PE10.3)
      CALL SYMBOL(X9,X9,PSIZE,AGAIN,D9,10)
      GOTO 540
530   CALL SYMBOL(0.3,0.,PSIZE,'PHASE ANGLE AXIS /PI RADIANS',D9,28)
540   CALL SYMBOL(0.3,5.0,PSIZE,'FREQUENCY*10',D9,12)
      FNUM=FLOAT(IBASE)
      CALL NUMBER(0.25,X9,0.07,FNUM,D9,-1)
      CALL SYMBOL(0.3,X9,PSIZE,' HZ',D9,3)
C
C  PLOT VALUES
C
      CALL PLOT(0.,0.,3)
      Y=0.0
      IF(J.EQ.1) IADR=0
      IF(J.EQ.2) IADR=2050
```

```
      IF(J.EQ.3) IADR=4100
      CALL NEWPEN(2)
      DO 570 K=1,NCMPLX
      CALL PLOT(ARRAY(IADR+K),Y,2)
 570  Y=Y+DFPLOT
C
C  END OF FRAME
C
 700  CONTINUE
C
C  END OF PLOT
C
 900  CALL PLOT(0.,0.,999)
C
C  CALL RASTER CONVERTION ROUTINE
C
      CALL RESET
      CALL CHAIN(PLTRSM,PARM,50)
      CALL EXIT
      END




C
C  THIS PROGRAM TAKES THE DATA FROM THE DISC FILE AND
C  CONVERTS IT TO RASTER FORMAT
C
C  WRITEN BY R.W.HOBBS 1985
C
      VIRTUAL SEIS(4156),IPBUF(25344),IBACK(25344)
      REAL*8 FSPECR,FSPECW,FSPECO,FSPECP,FSPECB
      REAL*4 PLTOUT(2),CONST(9),XMVT(4),PAR(5)
      INTEGER*2 ITR(5),PARM(50),HBLK(256)
C
      COMMON/PASS/ PARM
C
      EQUIVALENCE (PARM(1),FSPECP),(PARM(5),IPLOT),(PARM(7),ISTART),
     +(PARM(8),IGATHR),(PARM(9),ITRCIN),(PARM(10),NSMPIN),
     +(PARM(11),NDELAY),(PARM(12),NLENG),(PARM(13),TLENG),
     +(PARM(15),FSAMP),(PARM(17),IAGC),(PARM(18),NIBS),
     +(PARM(19),GAIN),(PARM(21),ICLIP),(PARM(22),NCLP),
     +(PARM(23),IPOL),(PARM(24),IREV),(PARM(25),ISDLY),
     +(PARM(26),ISAMPS),(PARM(39),FSPECW),(PARM(43),ITS),
     +(PARM(44),ITE)
      EQUIVALENCE (CONST(1),XMVT(1)),(CONST(5),PAR(1))
C
      DATA DRHN/3RDR /,APHN/3RAP /
      DATA FSPECO/12RDR4RASTERDAT/
      DATA PLTOUT(1)/6RDK1PLT/,PLTOUT(2)/6ROUTSAV/
C
C  GET CHAIN
C
      CALL RCHAIN(ICHFLG,PARM,50)
      IF(ICHFLG.NE.-1) STOP'THIS PROGRAM MAY ONLY BE CALLED BY PLOT'
C
C  SET UP I/O CALLS
C
      IF(IFETCH(DRHN).NE.0) STOP'DR FETCH ERROR'
      IF(IFETCH(APHN).NE.0) STOP'AP FETCH ERROR'
      CALL APINIT(0,0,IAP)
      IRD=IGETC()
      IWT=IGETC()
      IBK=IGETC()
      FSPECR=FSPECW
      IF(LOOKUP(IRD,FSPECR).LT.0) STOP'NO INPUT DATA FILE'
      FSPECB=FSPECP
      ILK=LOOKUP(IBK,FSPECB)
      IF(ILK.LT.0) STOP'NO BACKGROUND FILE'
C
C  SET UP CONSTANTS
```

```
C
      NSAMP=2048
      NSMPOT=IFIX(200.0*TLENG)
      NMK=NSAMP/ISAMPS
      NBLKR=NSMPIN/128
      IF(MOD(NSMPIN,128).NE.0) NBLKR=NBLKR+1
      IWORDR=2*NSMPIN
      NBLKW=99
      IWORDW=25344
      NBLKB=99
      IWORDB=25344
      IBKTOT=NBLKR*ITRCIN*IGATHR+1
      PWIDTH=FLOAT(IGATHR)*FLOAT(ITE-ITS+1)*FLOAT(NIBS)
      IF(IPLOT.EQ.0) PWIDTH=PWIDTH+(FLOAT(IGATHR-1)*5.0*FLOAT(NIBS))
      NBLKOT=IFIX((PWIDTH+500.0+(6.0*FLOAT(NIBS)))*132.0/256.0)+2
      IF(ILK.GT.NBLKOT) NBLKOT=ILK
      IBLKW=1
      IBLKB=1
      NTRACE=IGATHR*ITRCIN+1
      IF(IPLOT.EQ.0) NTRACE=NTRACE+((IGATHR-1)*5)
C
C  OPEN OUTPUT FILE AND PREPARE PLOTTER
C
      IF(IENTER(IWT,FSPECO,NBLKOT).LT.0) STOP'OUTPUT FILE ENTER ERR'
C
C  SET UP AP ADDRESES
C
      ITR(1)=0
      ITR(2)=ITR(1)+NSAMP
      ITR(3)=ITR(2)+NSAMP
      ITR(4)=ITR(3)+NSAMP
      ITR(5)=ITR(4)+NSAMP
      IA=ITR(5)+NSAMP
      IB=IA+4096
      IC=IB+4096
      IC1=IC+4096
      IC2=IC1+1
      IC3=IC2+1
      IC4=IC3+1
      IC5=IC4+1
      IC6=IC5+1
      IC7=IC6+1
      IC8=IC7+1
      IC9=IC8+1
      I1=IA
      I2=I1+NSAMP
      I3=I2+NSAMP
      I4=I3+NSAMP
      I5=I4+NSAMP
      I6=I5+NSAMP
      LEVOFF=23552
      ILEVND=LEVOFF+(4*NIBS)
C
C  READ HEADER BLOCK
C
      IF(IREADW(256,HBLK,0,IRD).NE.256) STOP'PLTDAT H/BLK READ ERR'
      IDELAY=HBLK(55)
      IF(NDELAY.LT.0) NDELAY=IDELAY
      NBEG=MIN0(IDELAY,NDELAY)
      XMVT(2)=(FLOAT(NLENG)*FSAMP)/FLOAT(NSMPOT)
      NSTART=IFIX(FLOAT(NDELAY-NBEG)*FSAMP/XMVT(2))
      JSTART=IFIX(FLOAT(IDELAY-NBEG)*FSAMP)
      IBLKR=1
      IF(IREV.EQ.1) IBLKR=IBKTOT-NBLKR
      IF(IREV.EQ.1) NBLKR=-NBLKR
C
C  SET UP CONSTANTS TABLE IN AP
C
      PAR(1)=-2.0*FLOAT(NIBS)
      PAR(2)=1.0
      PAR(3)=GAIN
      PAR(4)=FLOAT(NCLP)
```

```
      IF(IPOL.EQ.-2) PAR(4)=-PAR(4)
      XMVT(1)=0.0
      XMVT(3)=1.0
      XMVT(4)=FLOAT(MAX0(NSAMP,NSTART+NSMPOT,JSTART+NSMPIN))
      CALL VCLR(ITR(1),1,5*NSAMP)
      CALL APWAIT
      CALL APPUT(CONST,IC1,9,2)
      CALL APWAIT
      CALL VRAMP(IC5,IC6,LEVOFF,1,4*NIBS+1)
      CALL APWR
C
C  COPY PLOT HEADER TO PLOTTER AND OUTPUT FILE
C
      IF(IRDRXM(IBACK(1),IBK,IWORDB,IBLKB,0).LT.0) STOP
     +'PLTDAT READ ERR 1'
      IBLKB=IBLKB+NBLKB
      L=1
      IBR=1
      DO 100 I=1,500
      CALL APPTXM(IBACK(IBR),IA,132,1)
      IBR=IBR+132
      CALL APGTXM(IPBUF(L),IA,132,1)
      L=L+132
      CALL APWD
      IF(IBR.LT.25344) GOTO 120
      IWB=IRDRXM(IBACK(1),IBK,IWORDB,IBLKB,0)
      IBLKB=IBLKB+NBLKB
      IBR=1
 120  IF(L.LT.25344) GOTO 100
      IF(IWDRXM(IPBUF(1),IWT,IWORDW,IBLKW,0).LT.0)
     +STOP'PLTDAT WRITE ERR 1'
      IBLKW=IBLKW+NBLKW
      L=1
 100  CONTINUE
C
C  READ FIRST TRACE
C
      CALL IRDRXM(SEIS(1),IRD,IWORDR,IBLKR,1)
      IBLKR=IBLKR+NBLKR
C
C  START TRACE LOOP
C
      IPAD=0
      II=0
      JJ=ISTART-1
      DO 800 I=1,NTRACE+4
      IF(II.EQ.ITRCIN) II=0
      CALL VCLR(ITR(5),1,NSAMP)
      IF(IPAD.GE.0) GOTO 300
      IF(I.GT.NTRACE) GOTO 300
      IF(II.EQ.0) JJ=JJ+1
      II=II+1
      IF(IPLOT.NE.0) GOTO 200
      IF(II.EQ.ITRCIN) IPAD=5
 200  IF(IWAIT(IWT).NE.0) STOP'PLTDAT WRITE ERR 2'
      IF(IWAIT(IRD).NE.0) STOP'PLTDAT READ ERR 2'
      CALL APWAIT
      CALL APPTXM(SEIS(1),IA,NSMPIN,2)
      CALL APWAIT
      CALL IRDRXM(SEIS(1),IRD,IWORDR,IBLKR,1)
      IBLKR=IBLKR+NBLKR
      IF(II.LT.ITS.OR.II.GT.ITE) GOTO 800
C
C  APPLY EXTRA AGC IF REQUIRED
C
      IF(IAGC.EQ.0) GOTO 250
      CALL VSQ(IA,1,IB,1,NSMPIN)
      CALL VRNAVG(IB,IC,NSMPIN,50)
      CALL VFILL(IC+50,IC,1,50)
      CALL VFILL(IC+NSMPIN-51,IC+NSMPIN-51,1,50)
      CALL VSQRT(IC,1,IC,1,NSMPIN)
      CALL LVNOT(IC,1,IB,1,NSMPIN)
```

```
      CALL VADD(IB,1,IC,1,IC,1,NSMPIN)
      CALL VDIV(IC,1,IA,1,IA,1,NSMPIN)
 250  CALL VSMUL(IA,1,IC7,IA,1,NSMPIN)
C
C  RE SCALE
C
      CALL VCLR(IB,1,NSAMP)
      CALL NMO(IC1,IC2,IC3,IC4,JSTART,IA,NSMPIN,NSTART,IB,NSMPOT,
     +-1,0)
C
C  CLIP VALUES OUTSIDE WINDOW
C
      IF(ICLIP.EQ.0) GOTO 280
      CALL LVGT(IB,1,IC8,0,IA,1,NSAMP)
      IF(IPOL.EQ.-2) CALL LVNOT(IA,1,IA,1,NSAMP)
      CALL VMUL(IA,1,IB,1,IB,1,NSAMP)
 280  CALL VMOV(IB+2047,-1,ITR(5),1,NSAMP)
      CALL MAXMGV(ITR(5),1,IC9,NSAMP)
      CALL APWR
      CALL APGET(VMAX,IC9,1,2)
      CALL APWD
      VMAX=VMAX/200.0
      TYPE 2000,JJ,II,VMAX
 2000 FORMAT('+GATHER ',I5,', TRACE ',I3,', HAS MAX.DEFLECTION  OF ',
     +F6.3,' INCHES')
C
C  ADD SEC MARKS
C
 300  CALL VCLR(I6,1,NSAMP)
      CALL VFILL(IC6,I6+NSAMP-ISDLY,-ISAMPS,NMK)
      CALL VFILL(IC6,I6+NSAMP-1,1,1)
C
C  RASTER NEXT NIBS OF PLOT
C
      ILEV=ILEVND
      IST=1
      CALL VCLR(I5,1,NSAMP)
      IF(IWAIT(IRD).NE.0) STOP'PLTDAT READ ERR 3'
      DO 400 J=1,NIBS
      DO 310 K=IST,3
      IF(IPOL.EQ.-2) GOTO 310
      CALL LINE(ITR(K),ILEV,ILEV+1,I1,I2,I5,IPOL,NSAMP,1)
 310  ILEV=ILEV-NIBS
      DO 320 K=4,5
      IF(IPOL.EQ.2) GOTO 320
      CALL LINE(ITR(K),ILEV,ILEV+1,I1,I2,I5,IPOL,NSAMP,-1)
 320  ILEV=ILEV-NIBS
      CALL VOR(I6,1,I5,1,I5,1,NSAMP)
C
C  RASTER LOGICAL VECTOR
C
      CALL VCLR(I3,1,132)
      CALL RASTER(I5,I3,NSAMP)
C
C  MERGE BACKGROUND
C
      IF(IWB.LT.0) GOTO 380
      CALL APPTXM(IBACK(IBR),I4,132,1)
      IBR=IBR+132
      CALL APWAIT
      CALL LOR(I3,I4,I3,132)
      IF(IBR.LT.IWB) GOTO 380
      IF(IWAIT(IWT).NE.0) STOP'PLTDAT WRITE ERR 3'
      IWB=IRDRXM(IBACK(1),IBK,IWORDB,IBLKB,0)
      IBLKB=IBLKB+NBLKB
      IBR=1
C
C  OUTPUT FINISHED RASTER
C
 380  CALL APWR
      CALL APGTXM(IPBUF(L),I3,132,1)
      CALL VCLR(I5,1,NSAMP)
```

```
      L=L+132
      CALL APWAIT
      IF(L.LT.25344) GOTO 390
      IF(IWAIT(IWT).NE.0) STOP'PLTDAT WRITE ERR 4'
      CALL IWDRXM(IPBUF(1),IWT,IWORDW,IBLKW,1)
      IBLKW=IBLKW+NBLKW
      L=1
 390  ILEV=ILEVND+J-NIBS
 400  IST=2
C
C  MOVE TRACES DOWN ONE
C
      CALL VMOV(ITR(2),1,ITR(1),1,4*NSAMP)
      IPAD=IPAD-1
 800  CONTINUE
C
C  FLUSH OUT REMAINING PLOT
C
 810  IF(IWB.LT.0) GOTO 850
      CALL APPTXM(IBACK(IBR),IA,132,1)
      IBR=IBR+132
      CALL APGTXM(IPBUF(L),IA,132,1)
      L=L+132
      CALL APWD
      IF(IBR.LT.IWB) GOTO 820
      IWB=IRDRXM(IBACK(1),IBK,IWORDB,IBLKB,0)
      IBLKB=IBLKB+NBLKB
      IBR=1
 820  IF(L.LT.25344) GOTO 830
      IF(IWDRXM(IPBUF(1),IWT,IWORDW,IBLKW,0).LT.0)
     +STOP'PLTDAT WRITE ERR 5'
      IBLKW=IBLKW+NBLKW
      L=1
 830  GOTO 810
 850  IF(IWAIT(IWT).NE.0) STOP'PLTDAT WRITE ERR 6'
      IF(L.EQ.1) GOTO 900
      L=L-1
      IF(IWDRXM(IPBUF(1),IWT,L,IBLKW,0).LT.0)
     +STOP'PLTDAT WRITE ERR 7'
C
C  CLOSE FILES AND RETURN
C
 900  FSPECP=FSPECO
      CALL CLOSEC(IBK)
      CALL IFREEC(IBK)
      CALL CLOSEC(IRD)
      CALL IFREEC(IRD)
      CALL CLOSEC(IWT)
      CALL IFREEC(IWT)
      CALL RESET
C
C  CHAIN OUTPUT ROUTINE
C
      CALL CHAIN(PLTOUT,PARM,4)
      STOP'CALL PLTRSM'
      END
```

```
C
C  THIS PROGRAM TAKES THE SPOOLED OUTPUT DISC FILE FROM
C  PLOT SUITE AND PUTS IT OUT TO THE PLOTTER
C
C  WRITTEN BY R. W. HOBBS 1985
C
      REAL*8 FSPECR,FSPECW
      INTEGER*2 PARM(4),IPBUF1(8448),IPBUF2(8448)
      LOGICAL*1 NAME(14),BUF,DUM
C
      EQUIVALENCE (PARM(1),FSPECW)
C
      DATA DRHN/3RDR /,MTHN/3RMT /
C
C  CHECK FOR CHAIN CALL
C
      CALL RCHAIN(ICHFLG,PARM,4)
      IF(ICHFLG.NE.-1)
     +STOP'THIS PROGRAM MAY ONLY BE CALLED BY PLOT SUITE'
C
C  OPEN INPUT FILE
C
      IF(IFETCH(DRHN).NE.0) STOP'DR FETCH ERR'
      IF(IFETCH(MTHN).NE.0) STOP'MT FETCH ERR'
      FSPECR=FSPECW
      IDCH=IGETC()
      INFLG=0
      CALL R50ASC(12,FSPECR,NAME)
      IF(NAME(1).EQ.'M') INFLG=1
      CALL MTXSET
      CALL MTX(IBUF,0,1)
      CALL MWAIT
C
C  LOOKUP PLOT FILE
C
      IF(LOOKUP(IDCH,FSPECR).LT.0)STOP'LOOKUP ERR'
      CALL IWAIT(IDCH)
      IBLK=0
      IN1=IREADW(256,IPBUF1(1),0,IDCH)
      IBLK=IBLK+1
      IN1=IREADW(8448,IPBUF1(1),IBLK,IDCH)
      IBLK=IBLK+33
      IF(IN1.LT.-1) STOP'PLOT READ ERR1'
      BUF=.TRUE.
C
C  COPY FILE TO PRINTER
C
 60   BUF=.NOT.BUF
      CALL MWAIT
      CALL IWAIT(IDCH)
      IF(BUF.EQ..FALSE.) IN2=IREAD (8448,IPBUF2(1),IBLK,IDCH)
      IF(BUF.EQ..TRUE.) IN1=IREAD (8448,IPBUF1(1),IBLK,IDCH)
      IBLK=IBLK+33
      IF(IN1.LT.-1.OR.IN2.LT.-1) STOP'PLOT READ ERR2'
      IF(INFLG.EQ.1.AND.(IN1.EQ.-1.OR.IN2.EQ.-1)) GOTO 100
      IF(BUF.EQ..FALSE.) CALL MTX(IPBUF1,IN1,2)
      IF(BUF.EQ..TRUE.) CALL MTX(IPBUF2,IN2,2)
      IF(IN1.GE.0.AND.IN2.GE.0) GOTO 60
 100  CALL MWAIT
      CALL MTX(IBUF,-1,1)
      CALL CLOSEC(IDCH)
      CALL RESET
      STOP'PLOT SUCCESSFULLY COMPLETED'
      END
```

VELAN

1. VELAN PROGRAM SUITE OVERVIEW

2. PROGRAM DESCRIPTIONS

2.1 SEG-Y to Disc Transfer Program, VELYSM.

2.2 Velocity Filter Program, FILTVF.

2.3 Velocity Analysis Program, VELANL.

2.4.1 Semblance Contouring Program, VELCON.

2.4.2 Semblance Statistics Program, VELSTS.

2.5 Gather Plot Program, SEMPLT.

2.6 Vector to Raster Conversion Programs, VELTRD, VELRSL.

2.7 Plot File Output to Printer Program, VELPLT

3. DESCRIPTION OF THE DATA FILE PARAMETERS

4. INPUT REQUIREMENTS FOR CONTINUOUS VELOCITY ANALYSES FROM A VELOCITY TAPE.

1. VELAN SUITE PROGAM OVERVIEW

The velocity analysis suite of programs allows the complete processing of common-mid point gathers on a Durham SEG-Y tape to either a contoured semblence plots or a statistical distribution plot ouput to the Versatec plotter.

The suite is composed of eight separate programs chained together to form a single package. The chaining of the suite allows starting the velocity analysis process at several different points and all the subsequent programs will be called automatically without further operator intervention until the plot is output to the printer by the last routine. The control parameters entered at any entry point are sufficient for the execution of the whole process. All the main processing programs have an interactive parameter input routine if the relavent parameter file is unavailable. The programs are as follows; a * indetifies an entry point; OPT identifies an optional routine:

PROGRAM NAME	PARAMETER FILE NAME	OPTIONS		DESCRIPTION
VELYSM	DK6:VELYSM.DAT	*		Reads data from SEG-Y tape to disc file.
FILTVF	DK6:FILTVF.DAT	*	OPT	Velocity filter routine.
VELANL	DK6:VELANL.DAT	*		Velocity analysis and semblance calulation.
VELCON		*	OPT	Semblance contouring and annotation.
VELSTS		*	OPT	Semblance statistics and annotation.
SEMPLT				Input gather scaling and plotting.
VELTRD VELRSL				Modified vector to raster conversion software.
VELPLT		*		Output plot files to printer.

Section 2 describes the whole process from reading the SEG-Y tape to the final output plot, showing how the individual programs are linked together and the structure of the temporary data files created. Section 3 examines the control parameter file used in the running of the velocity analysis suite and the differances needed in the relavent parameter files when starting the process at one of the other entry points. Section 4 explains the setting up of a velocity data tape and command file for the repeated velocity analysis along a given seismic reflection profile.

Below are listed the program names and the relevent input and output devices or files for the different modes of operation. 'Chained' indicates the program has been called by the previous routine; 'para. file' means the file name is defined in the parameter file; 'inter.' means the file name is input in reply to questions on the terminal when using interactive mode; DK or DR specifies the disc unit used.

PROGRAM NAME	PARAMETER FILE NAME	INPUT DATA	OUTPUT DATA
VELYSM	DK6:VELYSM.DAT	TAPE	DR6:DATA.DAT
	interactive	TAPE	DR6:DATA.DAT
FILTVF	chained	DR6:DATA.DAT	DR5:FILTER.DAT
	DK6:FILTVF.DAT	para. file	DR5:FILTER.DAT
	interactive	inter.	DR5:FILTER.DAT
Velocities taken from:		DK6:VELFIL.DAT	
VELANL			
no velocity filter	chained	DR6:DATA.DAT	DR5:CONTUR.DAT
	DK6:VELANL.DAT	para. file	DR5:CONTUR.DAT
	interactive	inter.	DR5:CONTUR.DAT
with velocity filter	chained	DR5:FILTER.DAT	DR5:CONTUR.DAT
	DK6:VELANL.DAT	para. file	DR5:CONTUR.DAT
	interactive	inter.	DR5:CONTUR.DAT
VELCON	chained	DR5:CONTUR.DAT	DR4:PARM.BIN DR4:VECTR1.BIN
	interactive	DR5:CONTUR.DAT	DR4:PARM.BIN DR4:VECTR1.BIN
VELSTS	chained	DR5:CONTUR.DAT	DR4:PARM.BIN DR4:VECTR1.BIN
	interactive	DR5:CONTUR.DAT	DR4:PARM.BIN DR4:VECTR1.BIN
SEMPLT			
no velocity filter	chained	DR6:DATA.DAT	DR4:GPLOT.DAT
with velocity filter	chained	DR5:FILTER.DAT	DR4:GPLOT.DAT
VELTRD	chained	DR4:PARM.BIN DR4:VECTR1.BIN	DR4:PARM.BIN DR4:VECTR2.BIN
VELRSL	chained	DR4:PARM.BIN DR4:VECTR2.BIN	DR4:VPLOT.DAT
VELPLT	chained	DR4:GPLOT.DAT DR4:VPLOT.DAT	plotter
	interactive	DR4:GPLOT.DAT DR4:VPLOT.DAT	plotter

2.1 VELYSM

This is a utility program to read Durham SEG-Y tapes and output an RT-11 disc file. This disc file is not SEG-Y standard due to historical reasons as it is compatible with the data format used by M. Pouter when the system was first set up. Hence many programs written at that time are still usable, also the data file may be used as input to the PLOT program. The program may also be used as a means of transferring data from tape to disc without running the rest of the analysis suite.

On transfer, the program strips off the SEG-Y trace headers and rounds up the trace sample count to a multiple of 128, a disc block boundary, by padding the input traces with zero samples. The traces are then packed together into a disc file starting at block 1, the second block in the file, without individual trace headers. The first block, block 0, of the file contains the trace header from the first trace read with the following variables deleted; trace number within reel; original file number; original trace number; source receiver offset and trace number within common-mid point gather. If the fold of cover is not more than 48 traces then the disc header also contains an array indicating whether a particular trace within the disc file gather has been zeroed, and an array of the time to the start of data for each trace, i.e. any mute function applied in previous processing. The recording delay, the time from the shot instant to the start of the recorded data, in the SEG-Y trace headers is assumed to be correct and must be constant for all the traces transfered.

To improve the signal to noise ratio of the data prior to velocity analysis lateral summing of several ajacent common-mid point, CMP, gathers may be requested, the number of traces in the output file is maintained but each output trace represents the normalised sum of all the non-zero input traces with the given trace number. If lateral sumnation of several CMP gathers is performed then the CMP gather number is changed to be the mean of the CMP gather numbers summed.

If more than one summed CMP gather is to be analysed at the same time then the program extends the output file as neccesary. The extra gathers are written in the same compact form as explained above without any individual trace headers. If the fold of cover is less than 48 traces then the single disc header block arrays indicate whether a mute fuction has been applied and if a particular trace has been zeroed in all the CMP gathers read.

To use the program for the first time it is advisable to use the interactive option. This saves the need to create a parameter file, also if the program executes correctly the values input at the keyboard will have been saved in a disc file, DK6:VELYSM.NEW. This file may be used as the parameter file for the re-running of the program by copying the file to DK6:VELYSM.DAT with the

following command:

COPY DK6:VELYSM.NEW DK6:VELYSM.DAT

The saving the run time parameters is only provided by this program of the VELAN suite.

INTERACTIVE INPUT

To start the program to run interactively, first ensure there is no data file on DK6: of the name VELYSM.DAT, this is the data file the program looks to for the parameters. Use the command:

DIR DK6:VELYSM.DAT

If present either rename with another name or delete:

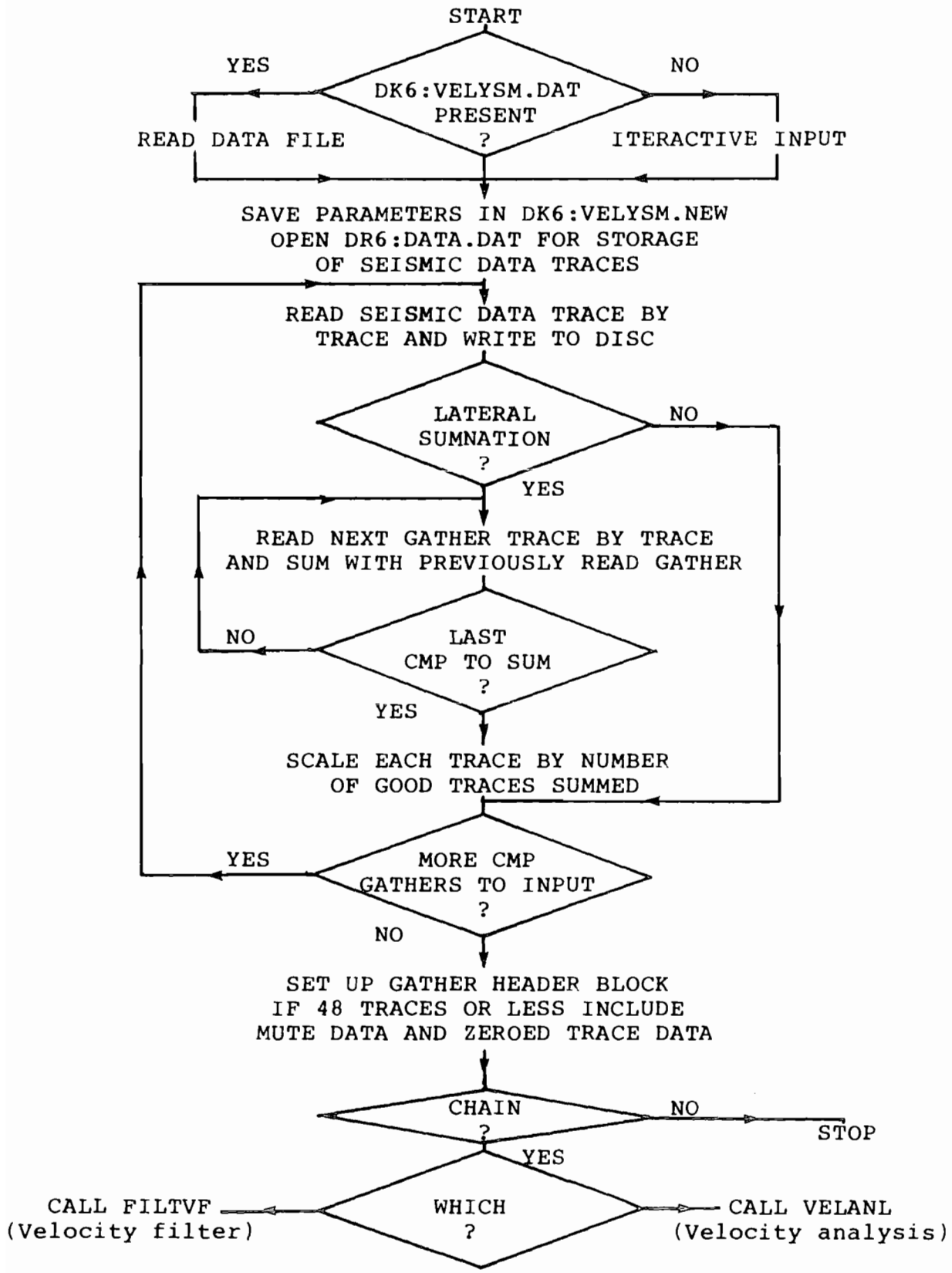
e.g. RENAME DK6:VELYSM.DAT DK6:VELYSM.OLD

or DELETE DK6:VELYSM.DAT

To start the program type:

RUN DK1:VELYSM

The program will prompt for parameters, reply following the format type indicated; INTEGER no decimal point; REAL with decimal point; (Y/N) either Y for yes or N for no. When asked for more than a single number on a given line seperate entries with a comma. End each reply, line or part line of numbers with a carriage return.



Flow Diagram for VELYSM

INTERACTIVE DESCIPTION

NO. OF TRACES IN I/P CMP is the fold of cover of the data.

NO. OF SAMPLES IN TRACE is given by the input trace or record length divided by the sampling interval. If this number does not equal the number recorded in the trace header the program will print an error message.

NO. OF GATHERS TO SUM, this is for lateral sumnation of sequential common-mid point gathers to improve signal to noise ratio.

NO. OF SUMMED GATHERS, this allows more than one velocity analysis to be done for each run of the program. Mainly for use with the statistics option, for normal use set to 1.

READ DRIVE NO. is the number of the tape drive on which the input tape is loaded.

CHAIN VELOCITY ANALYSIS SUITE, if the answer is no, the program will just copy the input gather onto disc and return the user to the PDP-11/34 operating system. If the reply is yes futher questions will be asked to assertain the processing parameters for the analysis.

OUTPUT PRESENTATION, this selects which output plot will be formed:

0 = normal contoured semblance plot, if the number of summed CMP gathers is greater than, one then a plot will be produced for each input gather.

1 = statistical plot, all the data from the input gathers will be presented on a single plot showing position of individual semblance maxima, position of mean semblance maxima, standard error about the mean and mean aspect ratio of the picked semblance peaks.

2 = summed semblance plot, the semblance data from each CMP analysis is summed to give a mean semblance function. The position of the maximum semblance and the aspect ratio for the peak is found for each time slice.

VELOCITY FILTER I/P GATHER this is an optional request that will add an extra routine to the analysis, that of velocity filtering (Ryu 1982). This is used

to suppress arrivals in the input gather that have a move-out correction greater than, or apparent velocity less than, a specified r.m.s. velocity time function. See section 2.2

The next two questions are only used if the velocity filtering option is used. See section 2.2 for a more complete description of the operation of this routine.

OPERATOR SIZE — this determines the efficiency of the process to suppress unwanted coherent arrivals in the common-mid point gather. The operator size must be odd, less than the fold of cover yet as large as possible to ensure effective filtering, up to a maximum of 29, if the given operator is too large the program will reduce the size automatically. A quick guide to acceptable operator sizes for a particular data set is that the operator size multiplied by the output sample count; end time of the analysis minus the start time of the analysis divided by the sample interval; must be less than 24000.

As the process needs to apply a move-out on the gather prior to filtering the r.m.s. velocity, time data is required. This data is stored in a disc file DK6:VELFIL.DAT, this file must be created using the interactive program VELDAT, see the Time Sequence Analysis Program documentation, prior to executing the velocity analysis suite.

% VELOCITY SHIFT — this determines what percentage of the defined velocity function to use for the move-out process. If DK6:VELFIL.DAT contains cut-off velocities then reply 100, if DK6:VELFIL.DAT contains primary velocity picks then a number less than 100 must be used; typically numbers in the range 75 to 90. If a number greater than 100 is entered the operator changes roll and will filter out coherent arrivals that have apparent velocities greater than the defined funtion

The program then issues a reminder that the cut-off velocity data must be loaded into a file named: DK6:VELFIL.DAT.

Whether the velocity analysis is run with or without velocity filtering the following questions will be asked.

SAMPLE INTERVAL — is the time in milliseconds between the data points in the input traces.

INITIAL, FINAL RMS VELOCITIES, this defines the velocity window over which the velocity analysis will be performed. The program expects two numbers in reply to this question. There must be at least 1 km/sec difference between the limits.

START TIME, END TIME — defines the two-way travel time window over which the analysis will be performed. This allows velocity analysis on only a subset of the data input giving better resolution on the time axis. The number of time steps is chosen by the program to be as near 256 steps as possible.

INITIAL OFFSET, XSTEP, — give two numbers that represent the source receiver spacing in metres to the first trace in the gather and the receiver spacing between the input traces also in metres.

The program then prompts if the user wishes to default the rest of the parameter file. This may be advisable for the first run of the program on a given data set. The final plot will indicate what parameters have been chosen also the input gather will be plotted with an AGC (automatic gain control) function applied so it is easier to determine whether a mute function is required or not.

DEFAULT OTHER PARAMETERS (Y/N) ?

The following parameters are defaulted:

INTERPOLATION LEVEL = 4

INPUT TRACE SCALING = NONE

MUTE FUNCTION — If a mute function has been applied in previous processing and the fold of cover is less than or equal to 48, the a correction will be applied to the semblance function for that mute. Otherwise no muting or correction are applied.

VELOCITY STEP = (FINAL VELOCITY - INITIAL VELOCITY)/120 + 1

LENGTH OF AVERAGING WINDOW = nearest power of 2 that is greater than:

4*(END TIME - START TIME)/256 milliseconds.

LENGTH OF TIME AXIS = 10 inches

MINIMUM SEMBLANCE CONTOUR = 0.2

SEMBLANCE CONTOUR INTERVAL = 0.1

If the user wishes to define all the other parameters the N, no, response will cause the following questions to be asked.

INTERPOLATION LEVEL gives the level to which the input data is re-sampled, the numbers allowable are 1,2 or 4. The effect is to divide the input sample interval by the interpolation factor, thus minimizing wavelet distortion introduced by the following processing. Typically values of 2 or 4 are used.

SCALING OF INPUT TRACE this allows weighting of the gather to the larger off-set traces, as these normally have less energy than the near off-set traces. This may give better velocity determination on coherent arrivals with small move-out.

The following question will only be asked if the fold of cover, i.e. the number of traces in the input gather is 48 or less.

MODIFY/INSERT MUTE FUNCTION, gives the option to either insert a mute function on a gather that has not had any applied in previous processing or to change the function if there has. In the case where a previous mute function is being modified the program automatically choses the more severe case. The use of this function is to mute out the start of traces, there by removing wide-angle reflections and refracted energy that may not conform to the hyperbolic move-out approximation.

If the answer is (Y)es, then a further question is asked for the mute values to be applied for each trace.

GIVE MUTE FUNCTION enter the absolute mute time in milliseconds for each trace in the gather in trace order. Twelve numbers per line separated by commas with a carriage return at the end of each line or part line.

The following questions will be asked whether a mute function is included or not.

VELOCITY STEP, this determines the resolution along the velocity axis. There is a limit on the velocity step size imposed by the program such that :

(FINAL-INITIAL VELOCITIES) / VELOCITY STEP < 120

LENGTH OF AVERAGING WINDOW this is the length in milliseconds of the Hanning widow averaging factor applied to the raw semblance data to present a smoothly varying semblence function to the contouring program.

LENGTH OF TIME AXIS the width of the plotting paper allows the time axis to be upto ten inches long, a number less than 10.0 to be entered.

MINIMUM CONTOUR & CONTOUR INTERVAL, semblance data always lies in the range 0.0 to 1.0, upto twenty contours may be requested but typically a minimum of 0.2 and interval of 0.1 is sufficient.

2.2 FILTVF

This routine may be called to remove strong coherent low apparent velocity arrivals from the input gather, e.g. sea water multiples. The technique employed is that described by Ryu (1982) of applying a false move-out correction to the gather such that primaries; first reflections from structure; are over corrected, and multiples are under corrected. If the data is then transformed into frequency wave number space, f-k domain, the two sets of arrivals seperate into different quadrants and hence the multiples may be supressed by a simple reject filter. As the filter is so simple it is computationally faster to transform the filter into the x-t domain and apply directly to the false moved-out gather by two dimensional convolution. The false move-out is then removed prior to writing the trace to the output file and calling the velocity analysis program.

If this routine was specified in VELYSM, see section 2.1, then it will be automatically called prior to the velocity analysis being done on the input gather. In this case the input gather will be always to have the name DR6:DATA.DAT, the output gather is always written to DR5:FILTER.DAT. Otherwise this routine may be called to filter a gather already stored on disc followed by the rest of the analysis suite, by typing:

RUN DK1:FILTVF

The program will first search for a parameter file DK6:FILTVF.DAT, see section 3. If this is not avaliable then the program will prompt for input from the terminal. The questions asked are a subset of those used for VELYSM, see section 2.1. Accept extra questions will be asked for the name of the input data file and if the recording delay in the gather header block, see section 2.1, is correct.

GIVE FULL NAME OF I/P FILE enter the full name of the file containing the input gather. Include all : and . .

USE INFORMATION IN HEADER BLOCK under all normal conditions the recording delay entered in the header block will be correct and should be used. This option allows for the case of data which has not been corrected for time break during any previous processing. If a new delay time to the start of traces is required reply no, N, and a supplimentary question will be asked for the new delay time.

The parameters entered interactively will not be saved in a disc file. See section 3 for description of parameter file DK6:FILTVF.DAT

The output gather will again be written to the file DR5:FILTER.DAT.

The velocity filter routine also needs information on the move-out velocity profile, this must be avaliable in a data file DK6:VELFIL.DAT. This has the same format as the velocity/time profile data files used in the Time Sequence Analysis Program, see section 4.1 of the documentation of that program. The file may be created interactively by using the program VELDAT, see above documentation section 4.2.

The file has the following structure:

```
VELOCITY ANALYSIS 1
VELOCITY ANALYSIS 2
    .       .
    .       .
```

The velocity analyses are entered in CMP order and have the following form:

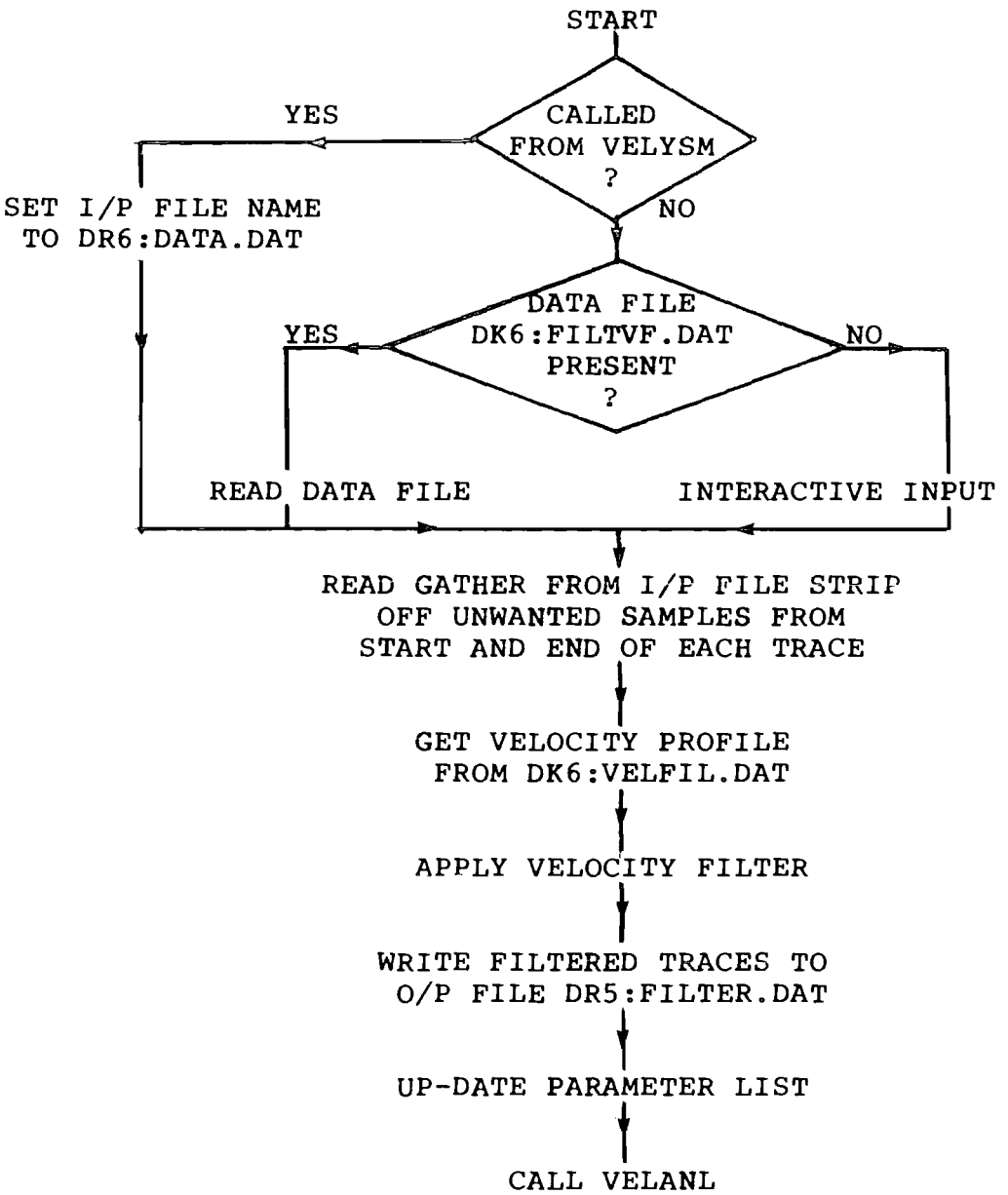
```
CMP NUMBER,  NUMBER OF LAYERS
TIME(1),  VELOCITY(1)
TIME(2),  VELOCITY(2)
  .           .
  .           .
```

CMP NUMBER (I6) Common mid-point number where given velocity function is to apply.

NUMBER OF LAYERS (I6) The number of velocity/time picks in the given analysis. Upto 10 layers may be entered.

TIME (1-N) (F10.0) The two-way travel time in milliseconds to a required pick.

VELOCITY (1-N) (F10.0) The r.m.s. velocity in kilometres/second at the given time.



Flow Diagram for the Velocity Filtering Program

2.3 VELANL

This part of the program suite performs the velocity analysis and semblance calculations. The program starts at the lowest velocity specified, reads each trace in turn, inserts a mute function if required, applies the normal move-out correction for the given velocity assuming hyperbolic trajectories, calculates the partial energy and the total stack energy for each sample within the given start and end times of the analysis. The semblance is then calculated from the following formula:

$$S_i = \frac{\left(\sum_{j=1}^{N} a_{i,j} \right)^2}{N * \sum_{j=1}^{N} \left(a_{i,j} \right)^2}$$

where i is the sample within trace
j is the trace number within gather
N is the total number of traces.

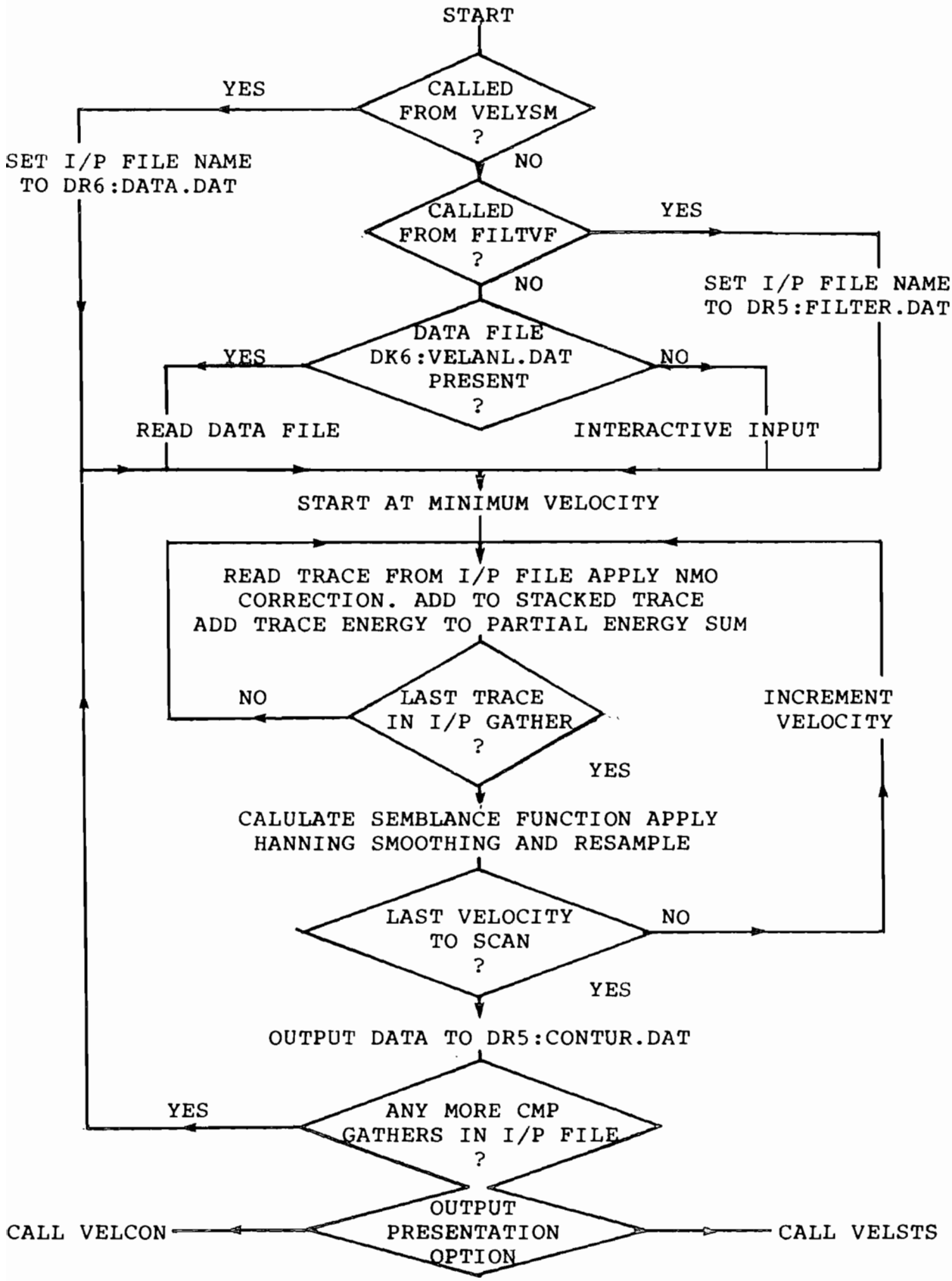
The resulting semblance function is then smoothed along the time axis by the application of a Hanning cosine function. As the function is weighted to the centre for a given window length this fuction gives a more accurate time determination than a simple linear average. After applying this function, the total number of semblance points is reduced to a number less than 256 per trace.

This process is repeated for each velocity step upto the last velocity specified. The semblance data is then written out to a data file, DR5:CONTUR.DAT, along with the velocity analysis scan parameters. If more CMP gathers are in the input file the data buffers are cleared and the process is run again. Control is then automatically passed onto the contouring package VELCON or the statistics package VELSTS depending on the output presentation option.

If the program has been called from either, the input routine, VELYSM, or the velocity filter routine, FILTVF, the correct gather data file will automatically be used. If running the program on a data file already stored on disc type:

RUN DK1:VELANL

The program will try to find a parameter file, DK6:VELANL.DAT, see section 3, if this is not present then interactive questions will be asked to assertain the required parameters. These questions will be a subset of those in section 2.1 as well as the extra two questions mentioned in section 2.2.



Flow Diagram for the Velocity Analysis Program VELANL

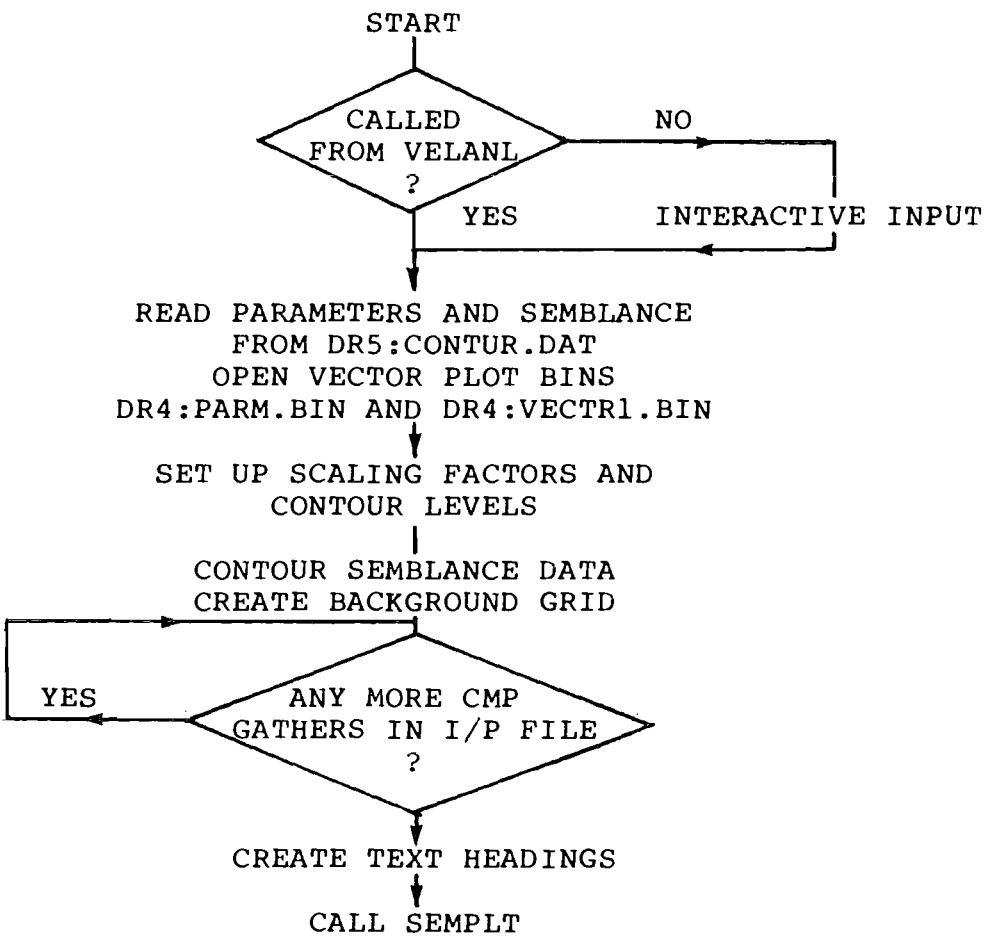
2.4.1 VELCON

The program reads the data file, DR5:CONTUR.DAT, determines the length of the plot and the contour levels for the semblance data. These values may either be passed from the calling program, VELANL, or if the program is run by typing:

```
RUN DK1:VELCON
```

the values may be entered interactively only. The semblance data is passed onto a contouring subroutine, this process normally takes three to four minutes depending on the number of contours to plot. The program creates a background grid for the contours scaled and labelled accordingly. To the right of the plot the peak semblance function is plotted for each time slice. A new map is made for each CMP gather input to the analysis. The rest of the parameters are incorporated in a heading and text panel to the side of the main plot.

Control is then passed to SEMPLT anlong with the file name of the input data to the velocity analysis program.



Flow Diagram for the Semblance Contouring Program VELCON

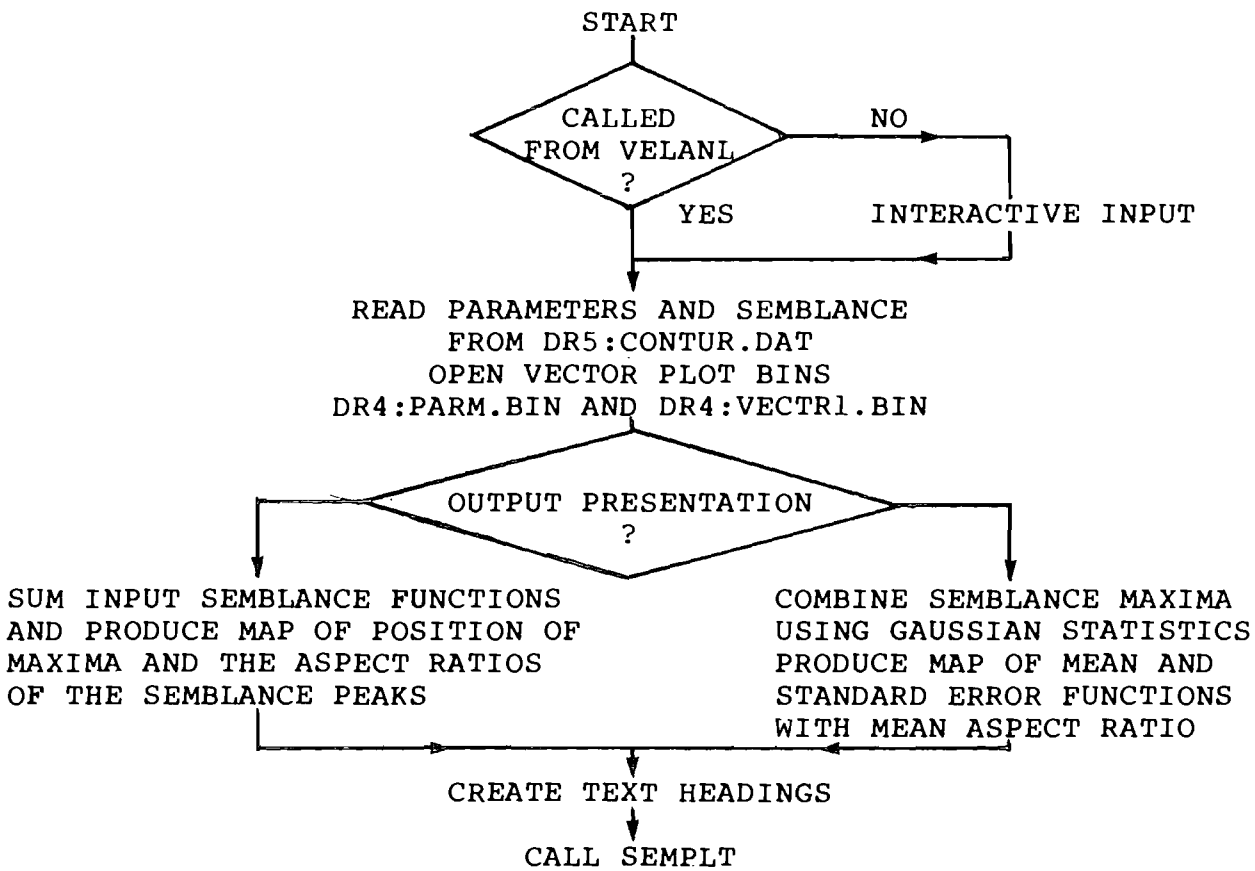
2.4.2 VELSTS

The program read the data file DR5:CONTUR.DAT to determine the plot parameters. This program may be run interactively if required:

RUN DK1:VELSTS

this allows the length of plot, minimum significant semblance value and output presentation to be changed without re-running the whole suite. Two types of plot are produced; the first treats each maxima from all the input plots as independent samples of a given velocity function, the mean, the standard error assuming a Gaussian distribution and the mean peak aspect ratios are calculated; the second sums all the input semblance fuctions and then picks the maximum of the averaged function for each time slice and calculates the aspect ratio of the peak.

The results are plotted on a time velocity grid with a heading and text panel for the input parameters. Control is passed to SEMPLT with the file name of the input data to the velocity analysis program.

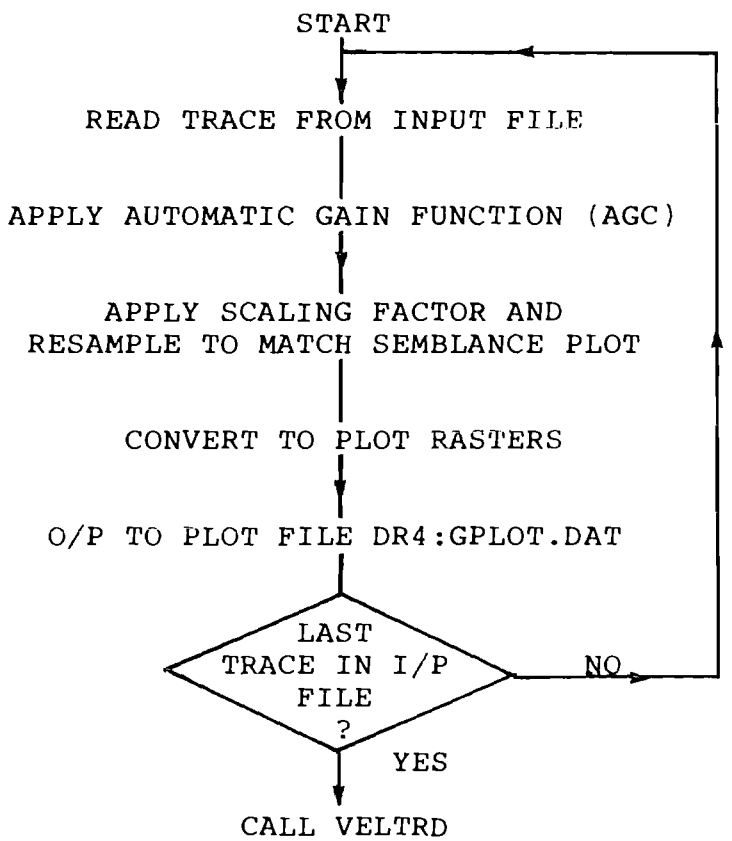


Flow Diagram for the Semblance Contouring Program VELSTS

2.5 SEMPLT

This is a high speed routine to output the gathers from which the semblance data has been calculated. The gather plot has an automatic gain function applied, scaled and resampled to the same length as the semblance plot. The traces are then converted to raster scans and output to a plot file, DR4:GPLOT.DAT.

The program cannot be called directly as it needs plotting parameters passed from the previous routine.



Flow Diagram for the Gather Plot routine SEMPLT

The plotted gather will show the data set as input to the velocity analysis routine, VELANL, however, it will not show the effect of any mute function applied during execution of this suite. When rescaling the time axis of the input traces to that of the analysis no check is made for aliasing, this may cause problems when reducing traces to more than two hundred samples per inch.

2.6 VELTRD and VELRSL

These are two routines taken from the software package supplied by Versatec Inc. for the conversion of vector files to a raster image that may be output to the plotter, THREAD and RASL. The programs have been modified specifically for the velocity analysis package as follows.

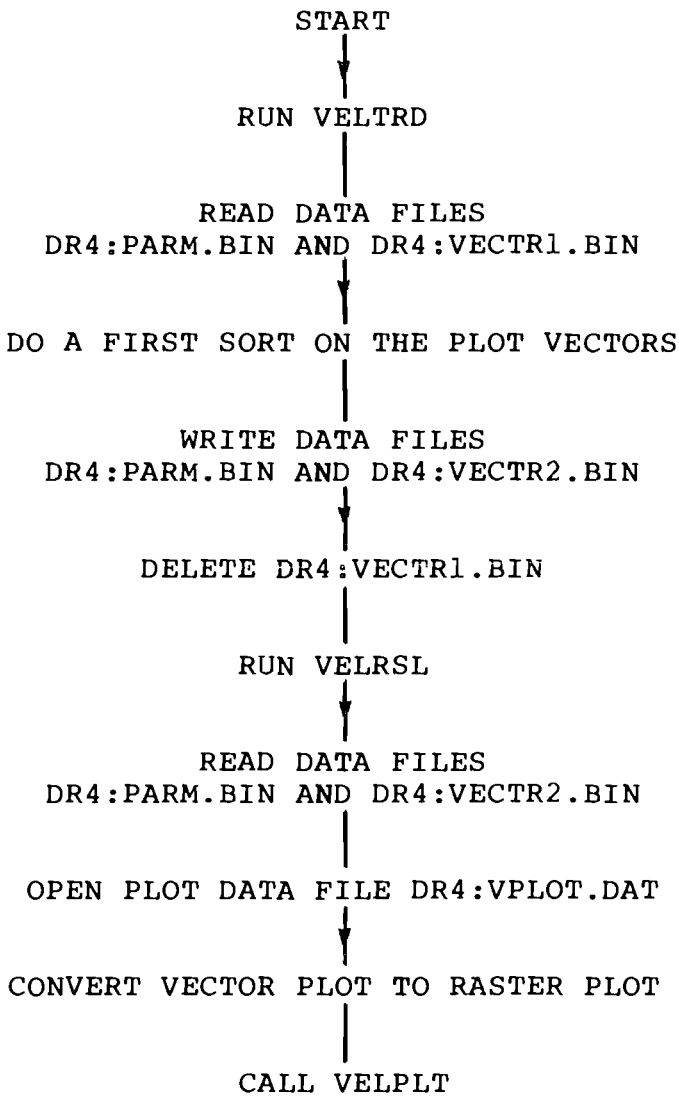
i) To allow the programs to be called from a previous routine and after execution to call a following routine.

ii) To output the raster image to a DR: disc file and not directly to the plotter. When the file is copied to the plotter, see section 2.7, this method gives a more even plot colour density.

The linked algorithm was selected, see Versatec documentation, as this allows a greater line density within a given plot than the more simple yet quicker mapped algorithm. The execution time of these pair of programs may be upto ten minutes depending on the number of lines to plot.

These two programs may not be directly called from the terminal as the vector plot files, DR4:PARM.BIN, DR4:VECTR1.BIN and DR4:VECTR2.BIN are modified and deleted by the programs as required.

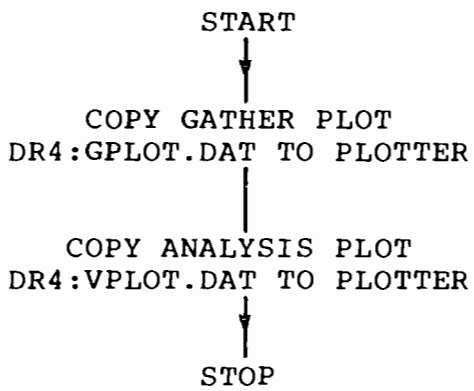
If there are a large number of semblance contours to plot then problems occur within these routines, manifestation of this are shown on the plot as the loss of some of the semblance peak information on the right of the plot. As it normally only effects the end of the data it has been ignored. Correction may be implimented, if required, by enabling the program to use more "active" lines, see Versatec documentation.



Flow Diagram for the Vector to Raster Conversion Programs
VELTRD and VELRSL

2.7 VELPLT

The final program of the velocity analysis suite copies both the plot files, DR4:GPLOT.DAT and DR4:VPLOT.DAT, to the Vesatec plotter. The gather is plotted prior to the analysis for the ease of cross referencing one to the other. The program may be run on its own with no interactive input necessary to obtain a second plot of a given analysis. Another copy of a velocity analysis may not be obtained by running the PLOT program, the programs use different file structures.



3. DESCRIPTION OF THE DATA FILE PARAMETERS

All the parameter files used in the velocity analysis suite have the same format, though not all the entries are present in every case as they depend on what program options are chosen.

A general description of the parameter file follows with examples of use at the end of the section.

```
FILE NAME
NTRAS,NSMPIN,IHFLG,NGATH,NANL,ICHFLG,NSUM,TPDRR
M,INDEN,INMUT,NOP,IVELSH
MUTE ( 1 to NTRAS )
VMIN,VMAX,VSTEP
TSTART,TEND,TSTEP,TGATE,TINDEL
XOFF,XSTEP
XLEN,CMIN,CINT
```

FILE NAME (14A1) — Used by the velocity filter routine and velocity analysis routine for the name of the input data disc file. This line must not be included for input to VELYSM.

NTRAS (I6) — Number of traces in input CMP gather.

NSMPIN (I6) — Number of samples per input trace. If running the suite from VELYSM then this number must be the exact number of samples per trace. If running the suite from FILTVF or VELANL then the number of input samples is rounded up to the next multiple of 128 samples.

IHFLG (I6) — Set to 1 if the recording delay in the header block is correct. In DK6:VELYSM.DAT this number is always 1, in other data files DK6:FILTVF.DAT and DK6:VELANL.DAT this number may be set to 0 or 1. If zero an extra entery is made on the time data line, TINDEL, to give the recording delay of the input traces.

NGATH (I6) — The number of summed CMP gathers to include in analysis. Normally set to 1 for contoured plot output but may be up to 20 for statistics plot.

NANL (I6) — Output presentation:

= 0 for contoured plot.

= 1 for statistical distribution plot.

= 2 for averaged semblance plot.

ICHFLG (I6) May be set to three possible options:

= 0 this indicates that only this program is to be executed.

= 1 this option calls the velocityanalysis routine.

= 2 used to call the velocity filter routine from VELYSM prior to calling the rest of the analysis suite.

In the FILTVF data file ICHFLG may only take the values of 0 or 1. When set to zero the input gather is velocity filtered only, the rest of the suite is not called. When set to 1, the normal value, after completing the filtering operation the gather is then passed onto the velocity analysis program. The next two entries may be omitted as they are not relavent.

In the VELANL data file the last three entries, ICHFLG, NSUM and TPDRR are not needed.

NSUM (I6) The number of ajacent input CMP gathers to sum. Normally set to 1.

TPDRR (I6) The input tape drive number, 0,1 or 2.

M (I6) The interpolation level for the input trace. This is used to re-sample the input data at a higher sampling frequency prior to normal move-out correction, this helps reduce distortion. The value of interpolation may be 1,2 or 4. The program checks that there is enough space for the interpolated trace, if not then it is automatically reduced and a warning message is set to the terminal.

INDEN (I6) This specifies the type of trace normalization to be done prior to the velocity analysis. It may take three values:

= -1 no normalization.

= 0 normalize each trace to unit r.m.s. energy.

= 1 normalize each trace to one over the total trace energy.

INMUT	(I6)	Indicates whether there is a mute function included in the data file. < = 0 no mute function in data file. = 1 mute fuction included in data file, see MUTE values. If there are more than 48 traces in the data file INMUT must be set to 0.

The next two entries are only used if the velocity filtering routine is called.

NOP	(I6)	The size of the velocity filter operator, must be odd and less than 29.
IVELSH	(I6)	The shift to be applied to the velocity function defined in DK6:VELFIL.DAT to give the cut-off velocity function used in velocity filtering, see section 2.2.
MUTE (1 to NTRAS)	(12I6)	If an input mute function is reqested, INMUT=1, then enter the mute values twelve per line in absolute milliseconds in input trace order.
VMIN	(F10.0)	The lower limit of the velocity scans in km/sec.
VMAX	(F10.0)	The upper limit of the velocity scans in km/sec.

The velocity limits VMIN and VMAX must be at least 1 km/sec apart.

VSTEP	(F10.0)	The velocity scan increment in km/sec.
TSTART	(F10.0)	The absolute start time for the analysis in milliseconds.
TEND	(F10.0)	The absolute end time of the analysis in milliseconds.
TSTEP	(F10.0)	The sampling interval of the input data in milliseconds.
TGATE	(F10.0)	The width of the Hanning smoothing window to be applied to the semblance data before plotting.
TINDEL	(F10.0)	The correct recording delay in milliseconds, only used if IHFLG is set to 0.

XOFF	(F10.0)	The source receiver offset in metres for the first trace in gather.
XSTEP	(F10.0)	The receiver spacing within gather in metres.
XLEN	(F10.0)	The length of the time axis in inches when plotted, if set to 0.0 the program sets the value to 10.0 inches. No value of greater than 10.0 inches may be entered.
CMIN	(F10.0)	The minimum semblance contour to plot. If set to 0.0 the program re-sets the value to 0.2.
CINT	(F10.0)	The interval between contours plotted. If 0.0 entered then the program takes a default value of 0.1

The semblance values always lie in the range 0.0 to 1.0, CMIN and CINT should be chosen so there are less than twenty contours.

CASE 1

Using VELYSM to transfer NGATH summed CMP gathers from tape to a disc file DR6:DATA.DAT.

The parameter file will have following entries:

```
NTRAS,NSMPIN,1,NGATH,0,0,NSUM,TPDRR
```

CASE 2

Copying a single gather from tape to disc with a velocity analysis and contoured plot output:

```
NTRAS,NSMPIN,1,1,0,1,1,TPDRR
M,INDEN,INMUT
If INMUT=1 then enter NTRAS mute values here
VMIN,VMAX,VSTEP
TSTART,TEND,TSTEP,TGATE
XOFF,XSTEP
XLEN,CMIN,CINT
```

CASE 3

As before with a velocity filter prior to velocity analysis:

```
NTRAS,NSMPIN,1,1,0,2,1,TPDRR
M,INDEN,INMUT,NOP,IVELSH
If INMUT=1 then enter NTRAS mute values here
VMIN,VMAX,VSTEP
TSTART,TEND,TSTEP,TGATE
XOFF,XSTEP
XLEN,CMIN,CINT
```

CASE 4

Statistical distribution plot of NGATH summed gathers with a velocity filter prior to velocity analysis:

```
NTRAS,NSMPIN,1,NGATH,1,2,NSUM,TPDRR
M,INDEN,INMUT,NOP,IVELSH
If INMUT=1 then enter NTRAS mute values here
VMIN,VMAX,VSTEP
TSTART,TEND,TSTEP,TGATE
XOFF,XSTEP
XLEN,CMIN,CINT
```

CASE 5

Averaged semblance plot of NGATH summed gathers with a velocity filter prior to velocity analysis:

```
NTRAS,NSMPIN,1,NGATH,2,2,NSUM,TPDRR
M,INDEN,INMUT,NOP,IVELSH
If INMUT=1 then enter NTRAS mute values here
VMIN,VMAX,VSTEP
TSTART,TEND,TSTEP,TGATE
XOFF,XSTEP
XLEN,CMIN,CINT
```

CASE 6

When running the velocity analysis suite on a gather already in a disc file, DR6:DATA.DAT, and extra line is added to the start of the parameter list giving the disc file name. N.B. NSMPIN must be rounded up to the next multiple of 128.

CASE 7

To change the recording delay for data already on disc set IHFLG to 0 and add new delay to time information line, e.g.

```
DR6:DATA.DAT
NTRAS,NSMPIN,0,NGATH,NANL,1,
M,INDEN,INMUT
If INMUT=1 then enter NTRAS mute values here
VMIN,VMAX,VSTEP
TSTART,TEND,TSTEP,TGATE,TINDEL
XOFF,XSTEP
XLEN,CMIN,CINT
```

CASE 8

To velocity filter gathers already on disc prior to analysis add the NOP and IVELSH parameters and set ICHFLG to 1.

```
DR6:DATA.DAT
NTRAS,NSMPIN,0,NGATH,NANL,1,1
M,INDEN,INMUT,NOP,IVELSH
If INMUT=1 then enter NTRAS mute values here
VMIN,VMAX,VSTEP
TSTART,TEND,TSTEP,TGATE,
XOFF,XSTEP
XLEN,CMIN,CINT
```

4. REPEATED PROGRAM EXECUTION FROM A VELOCITY TAPE

Either using the velocity gather output option in the sort program, DK1:SORT, or by using the time sequence analysis program, DK1:SEIS, and the tape positioning program, RHLOCY, copy the required gathers from the CMP sorted tapes to a seperate output tape. Do the any necessary processing using the Time Sequence Analysis Package and ensure the tape is terminated by a logical end of tape mark, use the program LEOT?, where ? is the tape drive number involved.

Run the velocity package interactively on the first gather or group of gathers, if lateral summing is to be used, and check the resultant plot is correct. If not repeat and modify the suspect parameters. The specified parameters will be stored in a parameter file DK6:VELYSM.NEW copy this file to DK6:VELYSM.DAT, see section 2.1.

Create a command file, e.g. DK6:VEL.COM, using the computer editor to read as follows:

```
RUN DK1:VELYSM
RUN DK1:VELYSM
 .       .
 .       .
 .       .
```

with as many RUN commands as necessary to finish the input data tape.

Rewind the input tape to the start and type:

```
@DK6:VEL.COM
```

This starts the process and it will repeat until either the data tape reaches the logical end of tape mark or the command file ends.

If two types of velocity analysis are required for each gather, e.g. with and without velocity filtering or examing different time windows. This may be done, in the first case, by copying DK6:VELYSM.DAT to DK6:FILTVF.DAT and with a simple edit command to delete the first character of the file, a line feed, and in its place insert the name of the output data file from VELYSM, DR6:DATA.DAT. Note: include : and . characters. Also add the operator size and velocity shift on the end of the third line. A velocity parameter file DK6:VELFIL.DAT must also be created using the program DK1:VELDAT.

The command file now has the following structure:

```
RUN DK1:VELYSM
RUN DK1:FILTVF
RUN DK1:VELYSM
RUN DK1:FILTVF
RUN DK1:VELYSM
RUN DK1:FILTVF
  .       .
  .       .
  .       .
```

The output will consist of the unfiltered gather followed by the filtered version.

For the second case, copy the data file DK6:VELYSM.DAT to DK6:VELANL.DAT. Again using the editor delete the first character and replace with the input data file name DR6:DATA.DAT, then change the time parameter line for the new time limits.

Set the command file to read:

```
RUN DK1:VELYSM
RUN DK1:VELANL
RUN DK1:VELYSM
RUN DK1:VELANL
RUN DK1:VELYSM
RUN DK1:VELANL
  .       .
  .       .
  .       .
```

Run the command file as before.

```
C
C   PROGRAM TO READ DURHAM SEG-Y FILES FROM TAPE,
C   PROVIDE OPTIONAL ADJACENT FILE SUMMING, AND
C   WRITE OUTPUT AS AN RT-11 VELOCITY FILE ON DISC.
C
C   D W WELLBY    11-MAR-83
C
C   MODIFIED BY R.W.HOBBS    FEB 1984
C
C
C     Added interactive input routine to VELYIN, this routine now either
C     reads parameters from the data file DK6:VELYSM.DAT if present or
C     prompt the user for information. The chosen parameters are then output
C     to DK6:VELYSM.NEW parameter file which by renaming as a DAT file can
C     be used to rerun the program.
C
C     Velocity filter routine added as a pre-analysis option as determined
C     by ICHFLG.
C
C     Added statistics package as an alternative method of looking at
C     the semblance data.
C
C     The whole suite consists of eight sub-programs chained together.
C     Parameters are passed via the chain call in an array PARM
C
C     COMMON PARAMETERS
C
C     TMNAM         radix 50 tape drive names
C     VELANL        name of velocity anaysis program for chain call
C     FILTVF        name of velocity filter program for chain call
C     PARM          parameter list pasted in chain call
C     TPDRR
C     TPDRW         tape drive names and error block
C     ITMERR
C     MUTE          array containing mute values, these are passed to the
C                   velocity analysis routine to obtain correct semblance
C                   values. They may be either taken from previous processing
C                   or entered at the time of anaysis. Where both are specified
C                   the routine will always pick the most severe.
C     OUTBUF        trace header buffer
C
C     NTRAS,NTRIN   number of traces in input gather
C     NSUM          number of gather summed prior to anaysis, improves S/N ratio
C     IWORDR        number of words per trace on tape
C     NSMPIN,       number of samples per input trace
C       NSAMP
C     INMUT         mute input flag
C     NGATH         number of input gathers in disc file, program will perform
C                   NGATH analyses, normally used with statistics option
C     XLEN          length in inches of time axis on final plot
C     CMIN          minimum semblance contour level or value for statistics
C     CINT          contour interval
C     TSTART        start time of analysis in msec.
C     TEND          end time of analysis in msec.
C     TGATE         length of Hanning smoothing window
C     LGATH         see NGATH
C     LANL          type of output plot 0=contoured
C                                       1=semb. max. scatter and statistics
C                                       2=statistics of summed semb. fuctions
C     FSPECO        disc file name
C     FSPECI        disc file name
C     TSTEP         sampling interval msec.
C     M             interpolation level for each trace prior to anaysis
C     INDEN         equalization energy flag -1=none
C                                            0=unit energy
C                                            1=inverse energy
C     VMIN          minimum velocity km/sec
C     VMAX          maximum velocity km/sec
C     VSTEP         velocity step km/sec, may be defaulted
C     XOFF          source receiver offset metres
C     XSTEP         trace spacing metres
C     NOP           operator size if velocity filtering
```

```
C     IVELSH      precentage shift to be applied to cut-off velocity function
C
      REAL*4 TMNAM(3),VELANL(2),FILTVF(2)
      INTEGER*2 PARM(50),TPDRR,TPDRW,ITMERR(2),MUTE(48)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/DATBUF/OUTBUF
      COMMON/TAPE/ TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/INPUT/ NTRAS,NSUM,IWORDR,NSMPIN,INMUT,MUTE,NGATH
      COMMON/VELANL/ PARM
C
      DATA TMNAM(1)/3RTM4/,TMNAM(2)/3RTM5/,TMNAM(3)/3RTM6/
      DATA APHN/3RAP /,DRHN/3RDR /,TMHN/3RTM /
      DATA VELANL(1)/6RDK1VEL/,VELANL(2)/6RANLSAV/
      DATA FILTVF(1)/6RDK1FIL/,FILTVF(2)/6RTVFSAV/
C
C  SET UP HANDLERS AND AP
C
      CALL RESET
      IF(IFETCH(APHN).NE.0) STOP'AP FETCH ERR'
      IF(IFETCH(DRHN).NE.0) STOP'DR FETCH ERR'
      IF(IFETCH(TMHN).NE.0) STOP'TM FETCH ERR'
      CALL APINIT(0,0,IAP)
C
C  GET INPUT
C
      CALL VELYIN(ICHFLG)
      ISTOP=0
C
C  DO TRANSFER
C
      CALL VELSUM(ISTOP,ICDPNO,ICHFLG)
      CALL VELEND(ISTOP,ICDPNO)
      CALL CLOSEC(TPDRR+10)
      CALL RESET
      IF(ISTOP.NE.0) GOTO 900
      IF(ICHFLG.EQ.1) CALL CHAIN(VELANL,PARM,50)
      IF(ICHFLG.EQ.2) CALL CHAIN(FILTVF,PARM,50)
  900 STOP
      END
```

```
      SUBROUTINE VELYIN(ICHFLG)
C
      REAL*8 FNAME,FSPECO,FSPECI
      REAL*4 TMNAM(3)
      LOGICAL*1 OUTBUF(240),RLHDBL(80,40),RLHDCD(400),IANS
      INTEGER*2 TPDRR,TPDRW,VCHN,PARM(50),MUTE(48),ITMERR(2)
C
      COMMON /DATBUF/OUTBUF
      COMMON /TAPE/TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON /INPUT/NTRAS,NSUM,IWORDR,NSMPIN,INMUT,MUTE,NGATH
      COMMON /VELANL/PARM
C
      EQUIVALENCE (PARM(1),XLEN),(PARM(3),CMIN),(PARM(5),CINT),
     +(PARM(7),TSTART),(PARM(9),TEND),(PARM(11),TGATE),
     +(PARM(13),LGATH),(PARM(14),LANL),(PARM(15),FSPECO),
     +(PARM(19),FSPECI),(PARM(23),NTRIN),(PARM(24),NSAMP),
     +(PARM(25),TSTEP),(PARM(29),M),(PARM(30),INDEN),
     +(PARM(31),VMIN),(PARM(33),VMAX),(PARM(35),VSTEP),
     +(PARM(37),XOFF),(PARM(39),XSTEP),(PARM(41),NOP),
     +(PARM(42),IVELSH)
C
      DATA FSPECO/12R             /
      DATA FNAME/12RDR6DATA  DAT/
      ICHFLG=0
      IHFLG=1
C
C  READ IN DATA FILE
C
      OPEN(UNIT=1,NAME='DK6:VELYSM.DAT',TYPE='OLD',ERR=10)
      GO TO 50
C
C  NO DATA FILE GO INTERACTIVE
C
 10   TYPE 500
 500  FORMAT('$ENTER NO. OF TRACES IN EACH I/P CMP GATHER',
     +' (INTEGER)? ')
      ACCEPT 1000,NTRAS
 11   TYPE 600
 600  FORMAT('$ENTER NO. OF DATA SAMPLES IN EACH I/P TRACE (INTEGER)? ')
      ACCEPT 1000,NSMPIN
      IF(NSMPIN.GT.4096) GOTO 11
      TYPE 700
 700  FORMAT('$ENTER NO. OF CMP GATHERS FOR LATERAL SUMMING ',
     +'(INTEGER)? ')
      ACCEPT 1000,NSUM
      IF(NSUM.EQ.0) NSUM=1
      TYPE 720
 720  FORMAT('$ENTER NUMBER OF SUMMED CMP GATHERS TO COPY TO',
     +' OUTPUT FILE (INTEGER)? ')
      ACCEPT 1000,NGATH
      TYPE 800
 800  FORMAT('$ENTER READ DRIVE NO. (INTEGER)? ')
      ACCEPT 1000,TPDRR
      TYPE 900
 900  FORMAT(/,'$CHAIN VELOCITY ANALYSIS SUITE (Y/N) ? ')
      ACCEPT 1100,IANS
 1100 FORMAT(A1)
      IF(IANS.EQ.'Y') ICHFLG=1
      IF(ICHFLG.EQ.0) GOTO 60
      TYPE 730
 730  FORMAT(' OUTPUT FORMAT 0 = CONTOURED PLOT',/,
     +       '                1 = PICKED MAXIMA SCATTER STATISTICS',/,
     +       '$               2 = SUMMED SEMBLANCE STATISTICS ? ')
      ACCEPT 1000,LANL
      TYPE 1200
 1200 FORMAT('$VELOCITY FILTER INPUT GATHER (Y/N) ? ')
      ACCEPT 1100,IANS
      IF(IANS.EQ.'Y') ICHFLG=2
      IF(ICHFLG.EQ.1) GOTO 40
C
C  COME HERE FOR VELOCITY FILTER INFORMATION
C
```

```
      TYPE 1250
 1250 FORMAT(/,'$GIVE VELOCITY FILTER OPERATOR SIZE,E.G. 11',
     +'(INTEGER)? ')
      ACCEPT 1000,NOP
      TYPE 1300
 1300 FORMAT(' GIVE % SHIFT FROM PICKED PRIMARY VELOCITY FUNCTION',/,
     +' TO GIVE A CUT-OFF VELOCITY FUNCTION BELOW WHICH COHERENT',/,
     +'$REFLECTORS WILL BE SUPPRESSED,E.G. 90 (INTEGER)? ')
      ACCEPT 1000,IVELSH
      TYPE 1350
 1350 FORMAT(' USES RMS VELOCITY DATA FILE DK6:VELFIL.DAT')
C
C  COME HERE FOR VELOCITY ANALYSIS DATA
C
 40   M=4
      INDEN=-1
      INMUT=-1
      TGATE=0.0
      XLEN=0.0
      CMIN=0.0
      CINT=0.0
C
      TYPE 1400
 1400 FORMAT(/,'$SAMPLING INTERVAL(MSEC) (REAL)? ')
      ACCEPT 2200,TSTEP
 2200 FORMAT(4F10.0)
      TYPE 1900
 1900 FORMAT('$INITIAL AND FINAL RMS VELOCITIES FOR ANALYSIS',
     +'(KM/SEC) (REAL)?,? ')
      ACCEPT 2200,VMIN,VMAX
      VSTEP=0.01*(VMAX-VMIN)
      IF(VSTEP.LE.0.0) STOP'VELOCITIES IN WRONG ORDER'
      TYPE 3000
 3000 FORMAT('$START AND END TIME FOR ANALYSIS(MSEC) (REAL)?,? ')
      ACCEPT 2200,TSTART,TEND
      TYPE 1950
 1950 FORMAT('$INITIAL SOURCE/RECEIVER OFFSET AND RECEIVER SPACING',
     +'(M) (REAL)?,? ')
      ACCEPT 2200,XOFF,XSTEP
      TYPE 1600
 1600 FORMAT('$DEFAULT ALL OTHER PARAMETERS (Y/N)? ')
      ACCEPT 1100,IANS
      IF(IANS.EQ.'Y') GOTO 60
C
C  IF NOT DEFAUTING ALL OTHER PARAMETERS ASK FOLLOWING QUESTIONS
C
      TYPE 1700
 1700 FORMAT(/,'$INTERPOLATION LEVEL FOR I/P DATA(1,2 OR 4)',
     +' (INTEGER)? ')
      ACCEPT 1000,M
      TYPE 1800
 1800 FORMAT(' SCALING OF INTERPOLATED TRACE ? ',/,
     %' LT.0 - NO SCALING',/,
     %' EQ.0 - SCALED TO UNIT ENERGY',/,
     %'$GT.0 - SCALED TO INVERSE ENERGY ')
      ACCEPT 1000,INDEN
      IF(NTRAS.GT.48) GOTO 45
      TYPE 1500
 1500 FORMAT('$MODIFY/INSERT TRACE MUTE FUNCTION (Y/N)?')
      ACCEPT 1100,IANS
      IF(IANS.EQ.'N') GOTO 45
      INMUT=1
      TYPE 1550
 1550 FORMAT(' GIVE MUTE FUNC. IN TRACE ORDER(MSEC)(INTEGER),',
     +' 12 PER LINE')
      ACCEPT 1000,(MUTE(I),I=1,NTRAS)
 45   TYPE 1920
 1920 FORMAT('$GIVE VELOCITY STEP SUCH THAT ((VMAX-VMIN)/VSTEP)<121',
     +' (KM/SEC)(REAL)? ')
      ACCEPT 2200,VSTEP
      TYPE 3100
 3100 FORMAT('$GIVE LENGTH OF AVERAGING WINDOW (MSEC) (REAL)? ')
```

```
      ACCEPT 2200,TGATE
C
C  GET DATA FOR RHVCON PROGRAM
C
      TYPE 3200
 3200 FORMAT('$LENGTH OF TIME AXIS IN INCHES (REAL)?' )
      ACCEPT 2200,XLEN
      TYPE 3300
 3300 FORMAT('$MINIMUM CONTOUR & CONTOUR INTERVAL (REAL)?,?')
      ACCEPT 2200,CMIN,CINT
      GOTO 60
C
C  READ DATA FILE DK6:VELYSM.DAT
C
 50   READ(1,1000) NTRAS,NSMPIN,IHFLG,NGATH,LANL,ICHFLG,NSUM,TPDRR
 1000 FORMAT(1216)
      IF(ICHFLG.EQ.0) GOTO 70
      IF(ICHFLG.EQ.1) READ(1,1000)M,INDEN,INMUT
      IF(ICHFLG.EQ.2) READ(1,1000)M,INDEN,INMUT,NOP,IVELSH
      IF(INMUT.GT.0.AND.NTRAS.GT.48) STOP
     +'NO MUTE ALLOWED IF INPUT GATHER HAS > 48 TRACES'
      IF(INMUT.GT.0) READ(1,1000) (MUTE(I),I=1,NTRAS)
      READ(1,2200)VMIN,VMAX,VSTEP
      READ(1,2200)TSTART,TEND,TSTEP,TGATE
      READ(1,2200)XOFF,XSTEP
      READ(1,2200)XLEN,CMIN,CINT
 70   CLOSE(UNIT=1)
C
C  SET UP CONSTANTS
C
 60   VCHN=14
      IF(NGATH.EQ.0) NGATH=1
      LGATH=NGATH
      NTRIN=NTRAS
      NSAMP=IFIX(NSMPIN/128.)*128
      IF(MOD(NSMPIN,128).NE.0) NSAMP=NSAMP+128
      IBLKV=NGATH*NTRAS*(NSAMP/128)+1
      IWORDR=NSMPIN*2+120
C
C  WRITE OUT DATA FILE AS DK6:VELYSM.NEW
C
      OPEN(UNIT=1,NAME='DK6:VELYSM.NEW',TYPE='NEW',ERR=74)
      GOTO 76
 74   STOP'DK6:VELYSM.NEW DATA FILE OPEN ERROR'
 76   WRITE(1,4100) NTRAS,NSMPIN,IHFLG,NGATH,LANL,ICHFLG,NSUM,TPDRR
      IF(ICHFLG.EQ.0) GOTO 78
      IF(ICHFLG.EQ.1) WRITE(1,4100) M,INDEN,INMUT
      IF(ICHFLG.EQ.2) WRITE(1,4100) M,INDEN,INMUT,NOP,IVELSH
      IF(INMUT.GT.0) WRITE(1,4100) (MUTE(I),I=1,NTRAS)
      WRITE(1,4200) VMIN,VMAX,VSTEP
      WRITE(1,4200) TSTART,TEND,TSTEP,TGATE
      WRITE(1,4200) XOFF,XSTEP
      WRITE(1,4200) XLEN,CMIN,CINT
 78   CLOSE(UNIT=1)
 4100 FORMAT(12(I5,','))
 4200 FORMAT(4(F9.3,','))
C
C  OPEN DISC FILE FOR RT-11 VELOCITY FILE
C
      FSPEC1=FNAME
      IER=IENTER(VCHN,FNAME,IBLKV)
      IF(IER.GT.0) GOTO 200
      STOP'VELOCITY DISC FILE ENTER ERR'
C
C  LOOKUP READ TAPE DRIVE
C
 200  ICHN=TPDRR+10
      ILK=LOOKUP(ICHN,TMNAM(TPDRR+1),-1)
      ISTUS=IPEEK("172520)
      IF((ISTUS.AND."100).GT.0) GOTO 80
      CALL CLOSEC(ICHN)
      TYPE 2000,TPDRR
```

```
 2000 FORMAT(' GIVEN READ DRIVE NO.:',I1,' NOT AVAILABLE',/,
     +'$GIVE NEW READ DRIVE NO.:')
      ACCEPT 1000,TPDRR
      IF(TPDRR.GE.3) STOP'NON-EXISTANT READ DRIVE'
      GO TO 200
C
C  IF LOOKUP OK, CHECK BOT FLAG
C
 80   IF((ISTUS.AND."40).LE.0) GOTO 110
C
C  READ REEL HEADERS FROM TAPE
C
      IF(IRMTLM(RLHDBL(1,1),ICHN,1600,ITMERR).NE.1600)
     $STOP'ASCII HDR READ ERR'
      CALL ISPFNW("376,ICHN)
      IF(IRMTLM(RLHDCD(1),ICHN,200,ITMERR).NE.200)
     $STOP'BINARY HDR READ ERR'
      CALL ISPFNW("376,ICHN)
 110  RETURN
      END




      SUBROUTINE VELSUM(ISTOP,ICDPNO,ICHFLG)
C
      VIRTUAL SEIS(4156)
      REAL*4 JCDPNO,JTRCNO,JFNUM,JTRAS,JCDPTR,DIVIDE(96)
      REAL*8 VELHD(64),VELDUM(30)
      REAL*4 TMNAM(3)
      INTEGER*2 TPDRR,TPDRW,ITMERR(2),VCHN,MUTE(48),MUTDUM(48),
     +IDEAD(48),PARM(50)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/DATBUF/OUTBUF
      COMMON/TAPE/TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
      COMMON/VELANL/ PARM
      COMMON/INPUT/NTRAS,NSUM,IWORDR,NSMPIN,INMUT,MUTE,NGATH
C
      EQUIVALENCE (OUTBUF(1),JTRCNO),(OUTBUF(9),JFNUM),
     $(OUTBUF(13),JTRAS),(OUTBUF(21),JCDPNO),(OUTBUF(25),JCDPTR),
     $(OUTBUF(29),ITYPE),(OUTBUF(35),INTR),(OUTBUF(109),IDELAY),
     $(OUTBUF(111),MUTST),(OUTBUF(113),MUTEND),(OUTBUF(115),NSMP),
     $(OUTBUF(1),VELDUM),(VELHD(31),MUTDUM(1)),(VELHD(43),IDEAD(1))
      EQUIVALENCE (PARM(24),NSAMP)
C
      VCHN=14
C
C  READ IN FIRST TRACE FROM TAPE
C
      ITRI=0
      DO 820 IG=1,NGATH
      ITRO=(IG-1)*NTRAS
      ITRI=ITRI+1
      CALL TRREAD(SEIS,ICDPNO,IWORDR,ITRI,0,ISTOP)
      IF(ISTOP.EQ.0) GOTO 10
      ICDPNO=ICDPNO+1
      RETURN
 10   IF(IG.GT.1) GOTO 57
      CALL JJCVT(JCDPNO)
      CALL IJCVT(JCDPNO,ICDPNO)
      ICDPDM=IFIX(ICDPNO+NSUM*NGATH*0.5)
      CALL JICVT(ICDPDM,JCDPNO)
      CALL JJCVT(JCDPNO)
      NDELAY=IDELAY
      ISTRT=1
      NSMP=NSAMP
      NSMPOT=NSMPIN
      LDELAY=NDELAY
      INTR=NTRAS
      IWORDV=NSMPOT*2
```

```
      IVBLK=IWORDV/256
      IF(MOD(IWORDV,256).NE.0) IVBLK=IVBLK+1
C
C BLANK INAPPLICABLE VARIABLES IN TRACE HEADER
C
      IBLANK=0
      CALL JICVT(IBLANK,JTRCNO)
      CALL JICVT(IBLANK,JFNUM)
      CALL JICVT(IBLANK,JTRAS)
      CALL JICVT(IBLANK,JCDPTR)
C
C  SAVE RT-11 256 WORD CDP GATHER HEADER BLOCK
C
      DO 55 I=31,64
 55   VELHD(I)=0.0
      DO 56 I=1,30
 56   VELHD(I)=VELDUM(I)
C
C  SET IDEAD TRACE VECTOR TO 'DEAD' TRACE FLAGS AS PER SEG-Y
C
      IF(NTRAS.GT.48) GOTO 57
      DO 54 I=1,NTRAS
 54   IDEAD(I)=2
C
C  CHECK TRACE IDENT CODE & BLANK 'DEAD' & 'DUMMY' TRACES
C
 57   IF(ITYPE.EQ.1) GOTO 62
      CALL VCLR(0,1,NSMPIN)
      CALL APGTXM(SEIS(61),0,NSMPIN,2)
      CALL APWD
 62   IF(IDELAY.EQ.NDELAY) GOTO 80
      ISTOP=26
      RETURN
C
C  WRITE OUT FIRST DATA TRACE
C
 80   IF(NTRAS.GT.48) GOTO 85
      MUTE(1)=MAX0(0,(MUTE(I)-LDELAY),(MUTEND-LDELAY))
      IF(ITYPE.EQ.1) IDEAD(1)=1
 85   ITRO=ITRO+1
      IBLK=(ITRO-1)*IVBLK+1
      IWDV=IWDRXM(SEIS(60+ISTRT),VCHN,IWORDV,IBLK,0)
      IF(IWDV.GE.IWORDV) GOTO 58
      ISTOP=22
      RETURN
 58   IF(NTRAS.EQ.1)GOTO 61
C
C  LOOP OVER ALL TRACES
C
      DO 60 I=2,NTRAS
      ITRI=ITRI+1
      CALL TRREAD(SEIS,ICDPNO,IWORDR,ITRI,0,ISTOP)
      IF(ISTOP.NE.0) RETURN
      IF(ITYPE.EQ.1) GOTO 59
      CALL VCLR(0,1,NSMPIN)
      CALL APWR
      CALL APGTXM(SEIS(61),0,NSMPIN,2)
      CALL APWD
 59   IF(NTRAS.GT.48) GOTO 86
      MUTE(I)=MAX0(0,(MUTE(I)-LDELAY),(MUTEND-LDELAY))
      IF(ITYPE.EQ.1) IDEAD(I)=1
 86   ITRO=ITRO+1
      IBLK=(ITRO-1)*IVBLK+1
      IF(IWDRXM(SEIS(60+ISTRT),VCHN,IWORDV,IBLK,0).GE.IWORDV) GOTO 60
      ISTOP=23
      RETURN
 60   CONTINUE
 61   CONTINUE
      IF(NSUM.EQ.1) GOTO 820
C
C  START SUMMING LOOP
C
```

```
      DO 100 I=1,96
 100  DIVIDE(I)=1.0
      DO 800 J=1,(NSUM-1)
      ITRO=(IG-1)*NTRAS
C
C  TRACE LOOP
C
      DO 700 I=1,NTRAS
      K=I-1
C
C  READ TRACE FROM TAPE AND PUT IN AP
C
      ITRI=ITRI+1
      CALL TRREAD(SEIS,ICDPNO,IWORDR,ITRI,0,ISTOP)
      IF(ISTOP.NE.0) RETURN
      IF(ITYPE.EQ.1) GOTO 740
      IF(J.LT.(NSUM-1)) GOTO 720
      CALL VCLR(0,1,NSMPIN)
      CALL APWR
      GOTO 730
 740  IF(IDELAY.EQ.NDELAY) GOTO 745
      ISTOP=26
      RETURN
 745  IF(NTRAS.GT.48) GOTO 735
      MUTE(I)=MAX0(0.,(MUTE(I)-LDELAY),(MUTEND-LDELAY))
      IDEAD(I)=1
 735  CALL APPTXM(SEIS(60+ISTRT),0,NSMPIN,2)
      CALL APWD
      DIVIDE(I)=DIVIDE(I)+1.0
C
C  READ ACCUMULATED TRACE SUM FROM DISC & PUT IN AP
C
 730  ITRO=ITRO+1
      IBLK=(ITRO-1)*IVBLK+1
      IF(IRDRXM(SEIS(61),VCHN,IWDV,IBLK,0).GE.0) GOTO 710
      ISTOP=24
      RETURN
 710  CALL APPTXM(SEIS(61),4096,NSMPOT,2)
      CALL APWD
      CALL VADD(0,1,4096,1,4096,1,NSMPOT)
      CALL APWR
      IF(J.LT.(NSUM-1)) GOTO 750
C
C  NORMALIZE TRACE
C
      CALL APPUT(DIVIDE,0,96,2)
      CALL APWD
 770  CALL VDIV(K,0,4096,1,4096,1,NSMPOT)
      CALL APWR
C
C  EXTRACT NEW PARTIAL SUM FROM AP & PUT BACK ON DISC
C
 750  CALL APGTXM(SEIS(61),4096,NSMPOT,2)
      CALL APWD
      IF(IWDRXM(SEIS(61),VCHN,IWDV,IBLK,0).GE.0) GOTO 720
      ISTOP=25
      RETURN
 720  IBLK=IBLK+IVBLK
 700  CONTINUE
 800  CONTINUE
 820  CONTINUE
C
C  WRITE OUT RT-11 256 WORD HEADER
C
 900  IF(NTRAS.GT.48) GOTO 855
      DO 850 I=1,NTRAS
 850  MUTDUM(I)=MUTE(I)
 855  IDELAY=LDELAY
      IF(IWRITW(256,VELHD(1),0,VCHN).EQ.256) GOTO 860
      ISTOP=21
      RETURN
 860  CALL CLOSEC(VCHN)
```

```
      RETURN
      END



      SUBROUTINE VELEND(ISTOP,ICDPNO)
C
      INTEGER*2 MUTE(48)
      LOGICAL*1 VELSUM(6),TRREAD(6)
C
      COMMON/INPUT/ NTRAS,NSUM,IWORDR,NSMPIN,INMUT,MUTE,NGATH
C
      CALL SCOPY('VELSUM',VELSUM,6)
      CALL SCOPY('TRREAD',TRREAD,6)
      IF(ISTOP.EQ.0) GOTO 500
      IF(ISTOP.GT.20) GOTO 100
      IF(ISTOP.GT.10) GOTO 200
 100  TYPE 1000,(VELSUM(I),I=1,6),ISTOP
 1000 FORMAT(/' TERMINATION FROM SUBROUTINE ',6A1,' ISTOP= ',I2)
      GOTO(21,22,23,24,25,26)(ISTOP-20)
 200  TYPE 1000,(TRREAD(I),I=1,6),ISTOP
      GOTO(11,12,13)(ISTOP-10)
 11   TYPE 2011
 2011 FORMAT(' TRACE READ ERROR - REWIND TAPE TO START OF GATHER
     + GROUP')
      GOTO 510
 12   TYPE 2012
 2012 FORMAT(' READ DRIVE EXIT')
      GOTO 510
 13   TYPE 2013
 2013 FORMAT(' REEL HEADER READ ERROR')
      GOTO 510
 21   TYPE 2021
 2021 FORMAT(' ERROR WRITING RT-11 DISC FILE HEADER - '/
     +' REWIND TAPE TO START OF GATHER GROUP')
      GOTO 510
 22   TYPE 2022
 2022 FORMAT(' ERROR WRITING TRACE TO DISC - '/
     +' REWIND TAPE TO START OF GATHER GROUP')
      GOTO 510
 23   GOTO 22
 24   TYPE 2024
 2024 FORMAT(' ERROR READING PARTIAL SUM TRACE FROM DISC - '/
     +' REWIND TAPE TO START OF GATHER GROUP')
      GOTO 510
 25   GOTO 22
 26   TYPE 2026
 2026 FORMAT(' ILLEGAL CHANGE IN TRACE DELAY BETWEEN SUMMED GATHERS')
      GOTO 510
 500  TYPE 5000,ICDPNO,(ICDPNO+NSUM*NGATH-1)
 5000 FORMAT(' CDP GATHER NOS.',I5,' TO',I5,
     +' SUCCESSFULLY ASSEMBLED AS VELOCITY FILE ON DR5',/)
      RETURN
 510  TYPE 5100,ICDPNO
 5100 FORMAT(' TERMINATION OCCURRED WHILE TRYING TO ASSEMBLE'/
     +' VELOCITY GROUP STARTING AT CDP GATHER NO.',I5)
      RETURN
      END
```

```
C  THIS PROGRAM DOES VELOCITY FILTERING BY 2-D CONVOLUTION
C  REF: RYU J.V. GEOPHYSICS VOL. 47 PP 869-883
C
C  WRITTEN BY R.W.HOBBS DECEMBER 1983
C
      REAL*8 FSPECR
      REAL*4 VELANL(2),T0LYR(10),VLYR(10)
      INTEGER*2 PARM(50),ICONST(3),MUTE(48)
C
      COMMON/FILE/ ICHFLG,IHFLG,IRD,IWT,NWDIN,NBLKIN,NWDOT,NBLKOT
      COMMON/DAT/ NSAMP,NTRIN,ISTRT,M,NSAMP2,NSAMOT,INMUT,MUTE,NGATH
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/TIME/ TINDEL,TSTEP,TSTART,TEND
      COMMON/FILT/ NOP,IVELSH,NVEL,ICONST,NF,ND
      COMMON/PASS/ PARM
      COMMON/VEL/ NLYR,T0LYR,VLYR
      COMMON/ADR/10,IA,IB,IC,IG,IX,IM,IV,IT,IS,IZ,LADR
C
      DATA APHN/3RAP /
      DATA DRHN/3RDR /
      DATA VELANL(1)/6RDK1VEL/,VELANL(2)/6RANLSAV/
      DATA MUTE(1)/48*0/
C
C  CHECK FOR CHAIN CALL
C
      CALL RCHAIN(ICHFLG,PARM,50)
C
C  SET UP I/O
C
      IF(IFETCH(APHN).NE.0) STOP'AP FETCH ERR'
      IF(IFETCH(DRHN).NE.0) STOP'DR FETCH ERR'
      CALL APINIT(0,0,IAP)
      IWT=11
      IRD=12
C
C  CHECK FOR DATA FILE OR CHAIN CALL OTHERWISE GO INTERACTIVE
C
      IF(ICHFLG.LT.0) GOTO 150
      OPEN(UNIT=1,TYPE='OLD',NAME='DK6:FILTVF.DAT',ERR=100)
      CALL INDKVF(FSPECR)
      CLOSE(UNIT=1)
      GOTO 200
  100 CALL INTTVF(FSPECR)
      GOTO 200
  150 CALL INCHVF(FSPECR)
C
C  GET VELOCITY FILE (IN FORM OF DK6:VELFIL.DAT)
C
  200 OPEN(UNIT=1,TYPE='OLD',NAME='DK6:VELFIL.DAT',ERR=300)
      GOTO 400
  300 STOP'NO VELOCITY FILE DK6:VELFIL.DAT'
  400 CONTINUE
C
C  SET UP CONSTANTS
C
      CALL SETVF(FSPECR,ICDPNO)
C
C  DO VELOCITY FILTERING
C
      CALL SUBVF(ICDPNO)
C
C  CLOSE DOWN AND CHAIN TO VELANL
C
      CALL CLOSEC(IRD)
      CALL CLOSEC(IWT)
      CLOSE(UNIT=1)
      CALL RESET
      TYPE 9000
 9000 FORMAT(' VELOCITY FILTERING SUCCESSFULLY COMPLETED',/)
      IF(ICHFLG.EQ.1) CALL CHAIN(VELANL,PARM,50)
      STOP
      END
```

```
      SUBROUTINE INTTVF(FSPECR)
C
C  INTERACTIVE INPUT SUBROUTINE FOR VELOCITY FILTERING PROGRAM
C
      REAL*8 FSPECR,FSPECI
      INTEGER*2 SPEC(39),EXT(4)
      LOGICAL*1 IANS,FBUF(16),IERR
      INTEGER*2 PARM(50),ICONST(3),MUTE(48)
C
      COMMON/FILE/ ICHFLG,IHFLG,IRD,IWT,NWDIN,NBLKIN,NWDOT,NBLKOT
      COMMON/DAT/ NSAMP,NTRIN,ISTRT,M,NSAMP2,NSAMOT,INMUT,MUTE,NGATH
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/TIME/ TINDEL,TSTEP,TSTART,TEND
      COMMON/FILT/ NOP,IVELSH,NVEL,ICONST,NF,ND
      COMMON /PASS/ PARM
C
      EQUIVALENCE (PARM(1),XLEN),(PARM(3),CMIN),(PARM(5),CINT),
     +(PARM(11),TGATE),(PARM(14),LANL),(PARM(30),INDEN),
     +(PARM(31),VMIN),(PARM(33),VMAX),(PARM(35),VSTEP)
      EQUIVALENCE (SPEC(16),FSPECI)
C
      DATA EXT/4*0/
C
      ICHFLG=0
C
C  GET INPUT FROM TERMINAL
C
      TYPE 400
 400  FORMAT(' USES RMS VELOCITY DATA FILE DK6:VELFIL.DAT',//,
     $'$GIVE FULL INPUT DATA FILE NAME ? ')
      CALL GETSTR(5,FBUF,14,IERR)
      IF(IERR.NE..FALSE.) STOP'ILLEGAL FILE NAME'
      IF(ICSI(SPEC,EXT,FBUF,,0).NE.0) STOP'FILE NAME ERROR'
      FSPECR=FSPECI
      TYPE 200
 200  FORMAT('$ENTER NUMBER OF CMP GATHERS IN INPUT FILE (INTEGER) ? ')
      ACCEPT 2100,NGATH
      TYPE 300
 300  FORMAT(' OUTPUT FORMAT 0 = CONTOURED PLOT',/,
     +       '               1 = PICKED MAXIMA SCATTER STATISTICS',/,
     +       '$              2 = SUMMED SEMBLANCE STATISTICS ? ')
      ACCEPT 2100,LANL
      TYPE 500
 500  FORMAT('$ENTER NO. OF TRACES IN EACH I/P CMP GATHER',
     +' (INTEGER)? ')
      ACCEPT 2100,NTRIN
 2100 FORMAT(12I6)
 11   TYPE 600
 600  FORMAT('$ENTER NO. OF DATA SAMPLES IN EACH I/P TRACE (INTEGER)? ')
      ACCEPT 2100,NSAMP
      TYPE 900
 900  FORMAT(/,'$CHAIN VELOCITY ANALYSIS SUITE (Y/N)?')
      ACCEPT 1100,IANS
 1100 FORMAT(A1)
      IF(IANS.EQ.'Y') ICHFLG=1
      TYPE 1250
 1250 FORMAT(/,'$GIVE VELOCITY FILTER OPERATOR SIZE,E.G. 11',
     +' (INTEGER)? ')
      ACCEPT 2100,NOP
      TYPE 1300
 1300 FORMAT(' GIVE % SHIFT FROM PICKED PRIMARY VELOCITY FUNCTION',/,
     +' TO GIVE A CUT-OFF VELOCITY FUNCTION BELOW WHICH COHERENT',/,
     +'$REFLECTORS WILL BE SUPPRESSED,E.G. 90 (INTEGER) ? ')
      ACCEPT 2100,IVELSH
C
C  GET TRACE INFORMATION AND SET DEFAULTS
C
      M=4
      INDEN=-1
      INMUT=-1
      TSTART=0.0
      TEND=0.0
```

```
      TGATE=0.0
      XLEN=0.0
      CMIN=0.0
      CINT=0.0
C
      TYPE 1400
 1400 FORMAT(/,'$SAMPLING INTERVAL(MSEC) (REAL)? ')
      ACCEPT 2200,TSTEP
      TYPE 1350
 1350 FORMAT('$USE HEADER BLOCK INFORMATION(Y/N) ? ')
      ACCEPT 1100,IANS
      IHFLG=1
      IF(IANS.EQ.'Y') GOTO 700
      IHFLG=0
      TYPE 1500
 1500 FORMAT('$INITIAL DELAY ON TRACES(MSEC) (REAL)? ')
      ACCEPT 2200,TINDEL
 2200 FORMAT(4F10.0)
 700  IF(ICHFLG.EQ.0) GOTO 800
      TYPE 1900
 1900 FORMAT('$INITIAL AND FINAL RMS VELOCITIES FOR ANALYSIS',
     +'(KM/SEC) (REAL)?,? ')
      ACCEPT 2200,VMIN,VMAX
      VSTEP=0.01*(VMAX-VMIN)
      IF(VSTEP.LE.0.0) STOP'VELOCITIES IN WRONG ORDER'
      TYPE 3000
 3000 FORMAT('$START AND END TIME FOR ANALYSIS(MSEC) (REAL)?,? ')
      ACCEPT 2200,TSTART,TEND
 800  TYPE 1950
 1950 FORMAT('$INITIAL SOURCE/RECEIVER OFFSET AND RECEIVER SPACING',
     +'(M) (REAL)?,? ')
      ACCEPT 2200,XOFF,XSTEP
      TYPE 1600
 1600 FORMAT('$DEFAULT ALL OTHER PARAMETERS (Y/N)? ')
      ACCEPT 1100,IANS
      IF(IANS.EQ.'Y') GOTO 9999
C
C  IF NOT DEFAUTING ALL OTHER PARAMETERS ASK FOLLOWING QUESTIONS
C
      TYPE 1700
 1700 FORMAT(/,'$INTERPOLATION LEVEL FOR I/P DATA(1,2 OR 4)',
     +'(INTEGER)? ')
      ACCEPT 2100,M
      IF(ICHFLG.EQ.0) GOTO 9999
      TYPE 1800
 1800 FORMAT(' SCALING OF INTERPOLATED TRACE ? ',/,
     %' LT.0 - NO SCALING',/,
     %' EQ.0 - SCALED TO UNIT ENERGY',/,
     %'$GT.0 - SCALED TO INVERSE ENERGY ')
      ACCEPT 2100,INDEN
      IF(NTRIN.GT.48) GOTO 950
      TYPE 1850
 1850 FORMAT('$MODIFY/INSERT TRACE MUTE FUNCTION (Y/N)?')
      ACCEPT 1100,IANS
      IF(IANS.EQ.'N') GOTO 950
      INMUT=1
      TYPE 1860
 1860 FORMAT(' GIVE MUTE FUNC. IN TRACE ORDER (MSEC)(INTEGER),',
     +' 12 PER LINE')
      ACCEPT 2100,(MUTE(I),I=1,NTRIN)
 950  TYPE 1920
 1920 FORMAT('$GIVE VELOCITY STEP SUCH THAT ((VMAX-VMIN)/VSTEP)<121',
     +' (KM/SEC)(REAL)? ')
      ACCEPT 2200,VSTEP
      TYPE 3100
 3100 FORMAT('$GIVE LENGTH OF AVERAGING WINDOW (MSEC) (REAL)? ')
      ACCEPT 2200,TGATE
C
C  GET DATA FOR RHVCON PROGRAM
C
      TYPE 3200
 3200 FORMAT('$LENGTH OF TIME AXIS IN INCHES (REAL)?' )
```

```
      ACCEPT 2200,XLEN
      TYPE 3300
 3300 FORMAT('$MINIMUM CONTOUR & CONTOUR INTERVAL (REAL)?,?')
      ACCEPT 2200,CMIN,CINT
C
C  RETURN
C
 9999 RETURN
      END




      SUBROUTINE INDKVF(FSPECR)
C
C  DISK INPUT SUBROUTINE FOR VELOCITY FILTERING PROGRAM
C
      REAL*8 FSPECR,FSPECI
      INTEGER*2 SPEC(39),EXT(4)
      LOGICAL*1 IANS,FBUF(16),IERR
      INTEGER*2 PARM(50),ICONST(3),MUTE(48)
C
      COMMON/FILE/ ICHFLG,IHFLG,IRD,IWT,NWDIN,NBLKIN,NWDOT,NBLKOT
      COMMON/DAT/ NSAMP,NTRIN,ISTRT,M,NSAMP2,NSAMOT,INMUT,MUTE,NGATH
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/TIME/ TINDEL,TSTEP,TSTART,TEND
      COMMON/FILT/ NOP,IVELSH,NVEL,ICONST,NF,ND
      COMMON /PASS/ PARM
C
      EQUIVALENCE (PARM(1),XLEN),(PARM(3),CMIN),(PARM(5),CINT),
     +(PARM(11),TGATE),(PARM(14),LANL),(PARM(30),INDEN),
     +(PARM(31),VMIN),(PARM(33),VMAX),(PARM(35),VSTEP)
      EQUIVALENCE (SPEC(16),FSPECI)
C
      DATA EXT/4*0/
C
      IHFLG=0
      ICHFLG=0
      TSTART=0.0
      TEND=0.0
C
C  READ DATA FILE
C
      CALL GETSTR(1,FBUF,14,IERR)
      IF(IERR.NE..FALSE.) STOP'ILLEGAL FILE NAME'
      IF(ICSI(SPEC,EXT,FBUF,,0).NE.0) STOP'FILE NAME ERROR'
      FSPECR=FSPECI
      READ(1,2100) NTRIN,NSAMP,IHFLG,NGATH,LANL,ICHFLG
 2100 FORMAT(12I6)
      IF(ICHFLG.EQ.0) READ(1,2100) M,NOP,IVELSH
      IF(ICHFLG.EQ.1) READ(1,2100) M,INDEN,INMUT,NOP,IVELSH
      IF(ICHFLG.EQ.0) GOTO 100
      IF(INMUT.GT.0.AND.NTRIN.GT.48) STOP
     +'NO MUTE ALLOWED IN I/P GATHER HAS > 48 TRACES'
      IF(INMUT.GT.0) READ(1,2100) (MUTE(I),I=1,NTRIN)
      READ(1,2200) VMIN,VMAX,VSTEP
      IF(IHFLG.EQ.0) READ(1,2200) TSTART,TEND,TSTEP,TGATE,TINDEL
      IF(IHFLG.GE.1) READ(1,2200) TSTART,TEND,TSTEP,TGATE
 2200 FORMAT(5F10.0)
      GOTO 200
  100 IF(IHFLG.EQ.0) READ(1,2200) TSTEP,TINDEL
      IF(IHFLG.GE.1) READ(1,2200) TSTEP
  200 READ(1,2200) XOFF,XSTEP
      IF(ICHFLG.EQ.1) READ(1,2200) XLEN,CMIN,CINT
      RETURN
      END
```

```
      SUBROUTINE INCHVF(FSPECR)
C
C  SUBROUTINE TO UNWRAP CHAINED INFORMATION
C  FOR VELOCITY ANALYSIS PROGRAM
C
C  R.W.HOBBS, APRIL 1983
C
      REAL*8 FSPECR,FSPECI
      INTEGER*2 PARM(50),ICONST(3),MUTE(48)
C
      COMMON/FILE/ ICHFLG,IHFLG,IRD,IWT,NWDIN,NBLKIN,NWDOT,NBLKOT
      COMMON/DAT/ NSAMP,NTRIN,ISTRT,M,NSAMP2,NSAMOT,INMUT,MUTE,NGATH
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/TIME/ TINDEL,TSTEP,TSTART,TEND
      COMMON/FILT/ NOP,IVELSH,NVEL,ICONST,NF,ND
      COMMON/PASS/ PARM
C
      EQUIVALENCE (PARM(7),RTSTRT),(PARM(9),RTEND),
     +(PARM(11),RTGATE),(PARM(13),LGATH),(PARM(19),FSPECI),
     +(PARM(23),NTRAS),(PARM(24),NSMPIN),(PARM(25),RTSTEP),
     +(PARM(27),RTNDEL),(PARM(29),IM),(PARM(30),IINDEN),
     +(PARM(31),RVMIN),(PARM(33),RVMAX),(PARM(35),RVSTEP),
     +(PARM(37),RXOFF),(PARM(39),RXSTEP),(PARM(41),LOP),
     +(PARM(42),LVELSH)
C
C  PASS PARAMETERS TO LOCAL VARIABLES
C
      IHFLG=1
      ICHFLG=1
      FSPECR=FSPECI
      NTRIN=NTRAS
      NSAMP=NSMPIN
      TSTART=RTSTRT
      TEND=RTEND
      TSTEP=RTSTEP
      TINDEL=RTNDEL
      NGATH=LGATH
      M=IM
      XOFF=RXOFF
      XSTEP=RXSTEP
      NOP=LOP
      IVELSH=LVELSH
C
C  RETURN
C
      RETURN
      END




      SUBROUTINE SETVF(FSPECR,ICDPNO)
C
C  SUBROUTINE TO CHECK AND SETUP CONSTANTS BEFORE FILTERING
C
      VIRTUAL RESULT(32)
      REAL*8 FSPECO,FSPECI,FSPECR
      REAL*4 X(8),VLYR(10),T0LYR(10),V2LYR(10),T02LYR(10),
     +VELINT(10),T0INT(10)
      INTEGER*4 JCDPNO
      INTEGER*2 PARM(50),ICONST(3),MUTDUM(48),MUTE(48),
     $HBLK(256),IVLYR(10),IT0LYR(10)
C
      COMMON/FILE/ ICHFLG,IHFLG,IRD,IWT,NWDIN,NBLKIN,NWDOT,NBLKOT
      COMMON/DAT/ NSAMP,NTRIN,ISTRT,M,NSAMP2,NSAMOT,INMUT,MUTE,NGATH
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/TIME/ TINDEL,TSTEP,TSTART,TEND
      COMMON/FILT/ NOP,IVELSH,NVEL,ICONST,NF,ND
      COMMON/PASS/ PARM
      COMMON/VEL/ NLYR,T0LYR,VLYR
      COMMON/ADR/ I0,IA,IB,IC,IG,IX,IM,IV,IT,IS,IZ,LADR
```

```
C
      EQUIVALENCE (PARM(7),RTSTRT),(PARM(9),RTEND),(PARM(13),LGATH),
     +(PARM(19),FSPECI),(PARM(23),NTRAS),(PARM(24),NSMPIN),
     +(PARM(25),RTSTEP),(PARM(27),RTNDEL),(PARM(29),LM),
     +(PARM(37),RXOFF),(PARM(39),RXSTEP),(PARM(41),LOP)
      EQUIVALENCE (HBLK(11),JCDPNO),(HBLK(55),IDELAY),
     +(HBLK(58),NSMP),(HBLK(100),ILYR),(HBLK(101),IT0LYR(1)),
     +(HBLK(111),IVLYR(1)),(HBLK(121),MUTDUM(1))
C
      DATA FSPECO/12RDR5FILTERDAT/
      INTSW=1
C
C  OPEN I/O DATA FILES
C
      ISIZE=LOOKUP(IRD,FSPECR)
      IF(ISIZE.LT.0) STOP'LOOKUP ERROR'
      IF(IENTER(IWT,FSPECO,ISIZE).LT.0) STOP'ENTER ERROR'
C
C  READ H/BLK
C
      IF(IREADW(256,HBLK,0,IRD).LT.0) STOP'H/BLK READ ERR'
      CALL JJCVT(JCDPNO)
      CALL IJCVT(JCDPNO,ICDPNO)
      CALL JJCVT(JCDPNO)
C
C  CHECK FOR REDUNDENT SAMPLES AT BOTH ENDS OF TRACE
C
      IF(IHFLG.EQ.1) TINDEL=FLOAT(IDELAY)
      NDELAY=IDELAY
      TBEG=TINDEL
      TLAST=TINDEL+(NSAMP*TSTEP)
      IF(TBEG.LE.TSTART) TBEG=TSTART
      TINDEL=TBEG
      IDELAY=IFIX(TINDEL)
      IF(TEND.EQ.0.0) TEND=TLAST
      IF(TEND.LE.TLAST) TLAST=TEND
      TRANGE=TLAST-TBEG
      NSAMOT=IFIX(TRANGE/TSTEP)
      IF(MOD(NSAMOT,128).NE.0) NSAMOT=128*(IFIX(NSAMOT/128)+1)
      NSMP=NSAMOT
      NBLKOT=NSAMOT/128
      NWDOT=NSAMOT*2
      NBLKIN=NSAMP/128
      NWDIN=NSAMP*2
      NSAMP2=2
  100 IF(NSAMP2.GE.NSAMOT) GOTO 200
      NSAMP2=NSAMP2*2
      GOTO 100
  200 FSAMP=1.0/TSTEP
      ISTRT=IFIX((TINDEL-FLOAT(NDELAY))*FSAMP)+1
C
C  OPTIMIZE AP USAGE
C
      IF(M.EQ.0) M=1
      IF(NOP.EQ.0) NOP=29
      NOP=(2*IFIX(NOP/2))+1
  300 RDAT=FLOAT(NSAMOT)*FLOAT(NOP)
      IF(RDAT.GT.24576.0) GOTO 450
  400 RSUM=RDAT+FLOAT(M*NSAMP2+2+NSAMOT+NOP*NOP+24)
      IF(RSUM.LE.24576.0) GOTO 500
      IF(M.EQ.1) GOTO 450
      M=M/2
      TYPE 2000,M
 2000 FORMAT(' **WARNING** INTERPOLATION LEVEL REDUCED TO:',I3)
      GOTO 400
  450 NOP=NOP-2
      TYPE 2010,NOP
 2010 FORMAT(' **WARNING** OPERATOR DIMENSION REDUCED TO:',I3)
      IF(NOP.LE.3) STOP'DATA TRACES TOO LONG'
      GOTO 300
  500 ICONST(1)=IFIX(TINDEL*FSAMP)
      ICONST(2)=ICONST(1)*M
```

```
      ICONST(3)=NSAMOT*M
C
C  UPDATE CHAIN PARAMETER LIST
C
      FSPEC1=FSPECO
      NSMPIN=NSAMOT
      NTRAS=NTRIN
      RTSTRT=TSTART
      RTEND=TEND
      RTNDEL=TINDEL
      RTSTEP=TSTEP
      LGATH=NGATH
      LM=M
      LOP=NOP
      RXOFF=XOFF
      RXSTEP=XSTEP
      NSMP=NSAMOT
C
C  SET UP AP ADDRESSES
C
      I0=0
      IA=NOP*NSAMOT
      IB=IA+M*NSAMP2+2
      IC=IB+NSAMOT
      IX=IC+NOP*NOP
      IG=IX-1
      IM=IX+1
      IV=IM+1
      IT=IV+10
      IS=IT+10
      IZ=24575
      LADR=IA-NSAMOT
C
C  LOAD OPERATOR
C
      CALL APWR
      CALL OPERVF(RESULT,NOP,IC,IVELSH)
      CALL APWR
C
C  READ FIRST VELOCITY ANALYSIS
C
 600  IVEL=1
      READ(1,1000)N2VAN,N2LYR
 1000 FORMAT(12I6)
      READ(1,1500)(T02LYR(I),V2LYR(I),I=1,N2LYR)
 1500 FORMAT(2F10.0)
      NLYR=N2LYR
      NVAN=N2VAN
      DO 710 I=1,NLYR
      T0LYR(I)=T02LYR(I)
      VLYR(I)=V2LYR(I)
      T0INT(I)=0.0
 710  VELINT(I)=0.0
      IDIFF=0
C
C SEE IF NEW VELOCITY ANALYSIS REQUIRED
C
      IF(ICDPNO.LT.N2VAN) GO TO 750
 720  NLYR=N2LYR
      NVAN=N2VAN
      DO 730 I=1,NLYR
      T0LYR(I)=T02LYR(I)
 730  VLYR(I)=V2LYR(I)
C
C READ IN NEXT ANALYSIS AND SET UP INTERPOLATION
C
      READ(1,1000,END=800)N2VAN,N2LYR
      READ(1,1500)(T02LYR(I),V2LYR(I),I=1,N2LYR)
      IF(ICDPNO.GE.N2VAN) GO TO 720
      IF(INTSW.EQ.0)GOTO 800
      INT=N2VAN-NVAN
      IF(INT.LE.1)GOTO 800
```

```
      RINT=FLOAT(INT)
      DO 740 I=1,NLYR
      T0INT(I)=(T02LYR(I)-T0LYR(I))/RINT
 740  VELINT(I)=(V2LYR(I)-VLYR(I))/RINT
      IF(ICDPNO.LE.NVAN) GO TO 800
      IDIFF=ICDPNO-NVAN
 750  IF(INTSW.EQ.0) GO TO 800
C
C ADD ON INTERPOLATING PARAMETERS
C
 760  DO 770 I=1,NLYR
      T0LYR(I)=T0LYR(I)+T0INT(I)
 770  VLYR(I)=VLYR(I)+VELINT(I)
      IF(IDIFF.LE.1) GO TO 800
      IDIFF=IDIFF-1
      GO TO 760
C
C  SAVE VELOCITY IN H/BLK
C
 800  IF(T0LYR(NLYR).GE.TEND) GOTO 810
      IF(NLYR.LT.10) GOTO 805
      T0LYR(NLYR)=TEND
      GOTO 810
 805  NLYR=NLYR+1
      T0LYR(NLYR)=TEND
      VLYR(NLYR)=VLYR(NLYR-1)
 810  IF(NTRIN.GT.48) GOTO 840
      DO 820 I=1,NTRIN
      IF(IHFLG.EQ.0) MUTDUM(I)=-1
      MUTE(I)=MAX0(0,MUTDUM(I)+NDELAY-IDELAY,MUTE(I)-IDELAY)
 820  MUTDUM(I)=MUTE(I)
 840  ILYR=NLYR
      DO 850 I=1,NLYR
      IT0LYR(I)=IFIX(T0LYR(I))
 850  IVLYR(I)=IFIX(10.0*VLYR(I)*FLOAT(IVELSH))
      IF(IWRITW(256,HBLK,0,IWT).LT.0) STOP'H/BLK WRITE ERR'
C
C  RETURN
C
      RETURN
      END




      SUBROUTINE SUBVF(ICDPNO)
C
C  VELOCITY FILTERING ROUTINE
C
      VIRTUAL SIN(4096),SOUT(4096)
      REAL*4 XMVT(23),VSAMP(10),TSAMP(10),VLYR(10),T0LYR(10)
      INTEGER*2 ICONST(3),MUTE(48)
C
      COMMON/FILE/ ICHFLG,IHFLG,IRD,IWT,NWDIN,NBLKIN,NWDOT,NBLKOT
      COMMON/DAT/ NSAMP,NTRIN,ISTRT,M,NSAMP2,NSAMOT,INMUT,MUTE,NGATH
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/FILT/ NOP,IVELSH,NVEL,ICONST,NF,ND
      COMMON/VEL/ NLYR,T0LYR,VLYR
      COMMON/ADR/ I0,IA,IB,IC,IG,IX,IM,IV,IT,IS,IZ,LADR
C
      EQUIVALENCE (XMVT(1),X),(XMVT(2),RM),(XMVT(3),VSAMP(1)),
     +(XMVT(13),TSAMP(1)),(XMVT(23),SH)
C
      ITRI=1
      ITRO=1
      CONST=FLOAT(NSAMOT+ICONST(1))
C
C  SET UP RMS VELOCITY / TIME TABLES IN SAMPLES
C
      DO 205 I=1,NLYR
      TSAMP(I)=FLOAT(IFIX(T0LYR(I)*FSAMP))
```

```
 205  VSAMP(I)=VLYR(I)/FSAMP
      IF(TSAMP(NLYR).LT.CONST) TSAMP(NLYR)=CONST
      RM=FLOAT(M)
      SH=0.01*FLOAT(IVELSH)
      CALL APWAIT
      CALL APPUT(XMVT(2),IM,22,2)
      CALL APWAIT
      CALL VSMUL(IV,1,IS,IV,1,10)
      CALL APWR
C
C  DO FILTER FOR EACH INPUT GATHER
C
      DO 900 LG=1,NGATH
      NF=(LG-1)*NTRIN
      ND=LG*NTRIN
      ITRI=ITRI-1
      ITRO=ITRO-1
C
C  LOAD AP WITH TRACES STARTING WITH A ZERO PAD OF (NOP-1)/2 TRACES
C
      LOOP=((NOP-1)/2)
      IADR=LOOP*NSAMOT
      CALL VCLR(I0,1,IADR)
      LOOP1=LOOP+1
      CALL DRAPVF(SIN,ITRI,IADR)
      ITRI=ITRI+1
      DO 500 I=1,LOOP1
      IF(ITRI.EQ.ND) GOTO 700
      CALL DRAPVF(SIN,ITRI,IADR)
      ITRI=ITRI+1
 500  IADR=IADR+NSAMOT
C
C  DO CONVOLUTION
C
 550  CALL VCLR(IA,1,NSAMOT)
      CALL CSQ2D(I0,1,IG,-1,IA,1,NSAMOT,NOP)
      CALL APWR
      CALL APDRVF(SOUT,ITRO,IB)
      ITRO=ITRO+1
      IF(ITRI.GT.ND) GOTO 600
      CALL DRAPVF(SIN,ITRI,LADR)
      ITRI=ITRI+1
      GOTO 550
C
C  COME HERE TO FLUSH OUT BUFFERS
C
 600  DO 650 I=1,LOOP
      CALL VCLR(LADR,1,NSAMOT)
      CALL CSQ2D(I0,1,IG,-1,IA,1,NSAMOT,NOP)
      CALL APWR
      CALL APDRVF(SOUT,ITRO,IB)
 650  ITRO=ITRO+1
      CALL APDRVF(SOUT,ITRO,IB)
      ITRO=ITRO+1
 900  CONTINUE
C
C  RETURN
C
      RETURN
 700  TYPE 9999
 9999 FORMAT(' TOO FEW TRACES TO FILTER')
      RETURN
      END
```

```
      SUBROUTINE DRAPVF(SIN,ITR,IADR)
C
C  SUBROUTINE TO TRANSFER DATA FROM DISK TO AP AND APPLY NMO
C
      VIRTUAL SIN(4096)
      INTEGER*2 ICONST(3),MUTE(48)
C
      COMMON/FILE/ ICHFLG,IHFLG,IRD,IWT,NWDIN,NBLKIN,NWDOT,NBLKOT
      COMMON/DAT/ NSAMP,NTRIN,ISTRT,M,NSAMP2,NSAMOT,INMUT,MUTE,NGATH
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/FILT/ NOP,IVELSH,NVEL,ICONST,NF,ND
      COMMON/ADR/ I0,IA,IB,IC,IG,IX,IM,IV,IT,IS,IZ,LADR
C
C  WAIT FOR READ/WRITE CHANNELS
C
      CALL IWAIT(IRD)
      CALL IWAIT(IWT)
C
C  IF FIRST CALL JUST READ TRACE FROM DISK
C
      IF(ITR.EQ.NF) GOTO 500
C
C  TRANSFER TRACE TO AP
C
      CALL APWAIT
      CALL VCLR(IA,1,M*NSAMP2+2)
      CALL APWAIT
      CALL APPTXM(SIN(ISTRT),IA,NSAMOT,2)
      X=XOFF+(FLOAT(ITR-NF-1)*XSTEP)
      CALL APWD
      CALL APPUT(X,IX,1,2)
      CALL APWD
C
C  TAPER BOTH ENDS OF TRACE
C
      CALL TAPER(IA,1,IA,1,10)
      CALL TAPER(IA+NSAMOT-1,-1,IA+NSAMOT-1,-1,10)
C
C  NOW INTERPOLATE TRACE
C
      CALL RFFT(IA,NSAMP2,1)
      CALL RFFTSC(IA,NSAMP2,3,1)
      CALL RFFTSC(IA,M*NSAMP2,-3,0)
      CALL RFFT(IA,M*NSAMP2,-1)
C
C  DO NMO
C
      CALL NMO(IX,IM,IV,IT,ICONST(2),IA,ICONST(3),ICONST(1),IADR,NSAMOT,
     $-1,0)
C
C  TRANSFER DATA FROM DISK TO MEMORY
C
 500  IF(ITR.EQ.ND) GOTO 900
      IBLK=ITR*NBLKIN+1
      CALL IRDRXM(SIN(1),IRD,NWDIN,IBLK,1)
C
C  RETURN
C
 900  RETURN
      END
```

```
      SUBROUTINE APDRVF(SOUT,ITR,IADR)
C
C  SUBROUTINE TO TRANSFER DATA FROM AP TO DISK AND APPLY OMN
C
      VIRTUAL SOUT(4096)
      INTEGER*2 ICONST(3),MUTE(48)
C
      COMMON/FILE/ ICHFLG,IHFLG,IRD,IWT,NWDIN,NBLKIN,NWDOT,NBLKOT
      COMMON/DAT/ NSAMP,NTRIN,ISTRT,M,NSAMP2,NSAMOT,INMUT,MUTE,NGATH
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/FILT/ NOP,IVELSH,NVEL,ICONST,NF,ND
      COMMON/ADR/ I0,IA,IB,IC,IG,IX,IM,IV,IT,IS,IZ,LADR
C
C  WAIT FOR AP
C
      IAD=IA+NSAMOT
      NCLR=(M*NSAMP2)-NSAMOT
      IAN=IA+(NOP/2)
      CALL APWAIT
C
C  IF FIRST CALL JUST READ TRACE FROM AP
C
      IF(ITR.EQ.NF) GOTO 500
C
C  TRANSFER DATA FROM MEMORY TO DISK
C
      CALL IWAIT(IRD)
      CALL IWAIT(IWT)
      IBLK=(ITR-1)*NBLKOT+1
      CALL IWDRXM(SOUT(1),IWT,NWDOT,IBLK,1)
C
C  TRACE TRANSFER FROM AP TO MEMORY
C
 500  IF(ITR.EQ.ND) GOTO 900
      X=XOFF+(FLOAT(ITR-NF)*XSTEP)
      CALL APPUT(X,IX,1,2)
      CALL APWD
      CALL VMOV(IAN,1,IA,1,NSAMOT)
      CALL VCLR(IAD,1,NCLR)
C
C  TAPER BOTH ENDS OF TRACE
C
      CALL TAPER(IA,1,IA,1,10)
      CALL TAPER(IAD-1,-1,IAD-1,-1,10)
C
C  NOW INTERPOLATE TRACE
C
      CALL RFFT(IA,NSAMP2,1)
      CALL RFFTSC(IA,NSAMP2,3,1)
      CALL RFFTSC(IA,M*NSAMP2,-3,0)
      CALL RFFT(IA,M*NSAMP2,-1)
C
C  DO NMO
C
      CALL OMN(IX,IM,IV,IT,ICONST(2),IA,ICONST(3),ICONST(1),IADR,NSAMOT)
C
C  GET OMN'ED DATA OUT OF AP
C
      CALL IWAIT(IWT)
      CALL APWR
      CALL APGTXM(SOUT(1),IADR,NSAMOT,2)
C
C  RETURN
C
 900  RETURN
      END
```

```
C
C    THIS PROGRAM PERFORMS
C    VELOCITY ANALYSES ON LARGE DATA SETS
C    BY CALCULATING SEMBLANCE ALONG HYPERBOLIC
C    TRAJECTORIES IN T-X SPACE
C    INTERPOLATION UP TO A LEVEL OF FOUR
C    IS PERMITTED FOR INPUT DATA
C
C    M.J.SMITH MAY 1982
C
C    R.W.HOBBS APRIL 1983
C              MODIFIED PROGRAM TO CHAIN VELCON OR VELSTS AND PLOT ROUTINES
C                       AND TO ALLOW CHAINING FROM TRACE SUM ROUTINE
C                       VELYSM
C                       TO TAKE INTO ACCOUNT ANY MUTE APPLIED IN
C                       IN PROCESSING OR ANY 'DEAD' TRACES
C                       IN APPLYING NORMALISATION TO SEMBLENCE VALUES
C
C    R.W.HOBBS DECEMBER 1983
C   CHANGED VELSUB TO MAKE USE OF HIGH SPEED NMO ROUTINE
C   CHANGED PROGRAM TO USE 48 TRACE DATA AND 4K SAMPLES
C
      REAL*8 FSPECR
      REAL*4 VELCON(2),VELSTS(2),DRHN,APHN
      INTEGER*2 PARM(50),MUTE(48),IHBLK(256),ICONST(3)
C
      COMMON/SETUP/ NSAMOT,NSTEP,NT0,ISTRT,IAPST,ISMPUT,NGATE,
     +N,IRES,NDEL
      COMMON/FILE/ IHFLG,IRD,NWDIN,NBLKIN,NVTOT,INDEN
      COMMON/DAT/ NSAMP,NTRIN,NDELAY,M,NSAMP2,NOP,NGATH,NANL
      COMMON/HEAD/ MUTE,IHBLK,ICONST
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/TIME/ TSTART,TEND,TSTEP,TGATE,TINDEL
      COMMON/VEL/ VMIN,VMAX,VSTEP
      COMMON/PASS/ PARM
      COMMON/ADR/ IA,IB,IC,ID,IE,IX,IM,IV,IT,IZ
C
      DATA APHN/3RAP /
      DATA DRHN/3RDR /
      DATA VELCON(1)/6RDK1VEL/,VELCON(2)/6RCONSAV/
      DATA VELSTS(1)/6RDK1VEL/,VELSTS(2)/6RSTSSAV/
      DATA MUTE(1)/48*0/
C
C  CHECK FOR CHAIN CALL
C
      CALL RCHAIN(ICHFLG,PARM,50)
C
C  SET UP I/O
C
      IF(IFETCH(APHN).NE.0)STOP'AP: FETCH ERROR'
      IF(IFETCH(DRHN).NE.0)STOP'DR: FETCH ERROR'
      CALL APINIT(0,0,IAP)
      IRD=12
      NOP=0
C
C  CHECK FOR DATA FILE OR CHAIN CALL OTHERWISE GO INTERACTIVE
C
      IF(ICHFLG.LT.0) GOTO 150
      OPEN(UNIT=1,TYPE='OLD',NAME='DK6:VELANL.DAT',ERR=100)
      CALL INDKVA(FSPECR)
      CLOSE(UNIT=1)
      GO TO 200
  100 CALL INTTVA(FSPECR)
      GOTO 200
  150 CALL INCHVA(FSPECR)
C
C  GET INPUT FILE
C
  200 IF(LOOKUP(IRD,FSPECR).LT.0)STOP 'LOOKUP ERR'
C
C  SET UP CONSTANTS AND OPEN OUTPUT FILE
C
```

```
      CALL VELSET
      ISIZE=2*(NVTOT+1)*NGATH+1
      OPEN(UNIT=2,TYPE='NEW',NAME='DR5:CONTUR.DAT',
     +INITIALSIZE=ISIZE,FORM='UNFORMATTED',ERR=999)
C
C  DO VELOCITY ANALYSIS
C
      CALL VELSUB(FSPECR)
C
C  CLOSE DOWN AND CHAIN TO VELCON
C
      CALL CLOSEC(IRD)
      CALL IFREEC(IRD)
      CLOSE(UNIT=2)
      CALL RESET
      TYPE 9000
 9000 FORMAT(' VELOCITY ANALYSIS SUCCESSFULLY COMPLETED',/)
      PARM(7)=NANL
      IF(NANL.NE.0) CALL CHAIN(VELSTS,PARM,7)
      CALL CHAIN(VELCON,PARM,6)
 999  STOP'DR5:CONTUR.DAT OPEN ERROR'
      END




      SUBROUTINE INTTVA(FSPECR)
C
C   INPUT PARAMETER SUBROUTINE FOR
C   VELOCITY ANALYSIS PROGRAM
C
      REAL*8 FSPECR,FSPECI
      INTEGER*2 PARM(50),MUTE(48),IHBLK(256),ICONST(3),
     +SPEC(39),EXT(4)
      LOGICAL*1 IANS,FBUF(16),IERR
C
      COMMON/FILE/ IHFLG,IRD,NWDIN,NBLKIN,NVTOT,INDEN
      COMMON/DAT/ NSAMP,NTRIN,NDELAY,M,NSAMP2,NOP,NGATH,NANL
      COMMON/HEAD/ MUTE,IHBLK,ICONST
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/TIME/ TSTART,TEND,TSTEP,TGATE,TINDEL
      COMMON/VEL/ VMIN,VMAX,VSTEP
      COMMON/PASS/ PARM
C
      EQUIVALENCE (PARM(1),XLEN),(PARM(3),CMIN),(PARM(5),CINT)
      EQUIVALENCE (SPEC(16),FSPECI)
C
      DATA EXT/4*0/
C
C  READ DATA FROM TERMINAL
C
      TYPE 100
 100  FORMAT('$GIVE FULL INPUT FILE NAME ? ')
      CALL GETSTR(5,FBUF,14,IERR)
      IF(IERR.NE..FALSE.) STOP'ILLEGAL FILE NAME'
      IF(ICSI(SPEC,EXT,FBUF,,0).NE.0) STOP'FILE NAME ERROR'
      FSPECR=FSPECI
      TYPE 200
 200  FORMAT('$ENTER NUMBER OF CMP GATHERS IN I/P FILE (INTEGER) ? ')
      ACCEPT 2100,NGATH
      IF(NGATH.EQ.0) NGATH=1
      TYPE 300
 300  FORMAT(' OUTPUT FORMAT  0 = CONTOURED PLOT',/,
     +       '                1 = PICKED MAXIMA SCATTER STATISTICS',/,
     +       '$               2 = SUMMED SEMBLANCE STATISTICS ? ')
      ACCEPT 2100,NANL
      TYPE 500
 500  FORMAT('$ENTER NO. OF TRACES IN EACH I/P CMP GATHER',
     +' (INTEGER)? ')
      ACCEPT 2100,NTRIN
 2100 FORMAT(1216)
```

```
 11   TYPE 600
 600  FORMAT('$ENTER NO. OF DATA SAMPLES IN EACH I/P TRACE (INTEGER)? ')
      ACCEPT 2100,NSAMP
C
C  GET TRACE INFORMATION AND SET DEFAULTS
C
      M=4
      INDEN=-1
      INMUT=-1
      TSTART=0.0
      TEND=0.0
      TGATE=0.0
      XLEN=0.0
      CMIN=0.0
      CINT=0.0
C
      TYPE 1400
 1400 FORMAT(/,'$SAMPLING INTERVAL(MSEC) (REAL)? ')
      ACCEPT 2200,TSTEP
      TYPE 1350
 1350 FORMAT('$USE HEADER BLOCK INFORMATION(Y/N) ? ')
      ACCEPT 1100,IANS
 1100 FORMAT(A1)
      IHFLG=1
      IF(IANS.EQ.'Y') GOTO 700
      IHFLG=0
      TYPE 1500
 1500 FORMAT('$INITIAL DELAY ON TRACES(MSEC) (REAL)? ')
      ACCEPT 2200,TINDEL
 2200 FORMAT(4F10.0)
 700  TYPE 1900
 1900 FORMAT('$INITIAL AND FINAL RMS VELOCITIES FOR ANALYSIS',
     +'(KM/SEC) (REAL)?,? ')
      ACCEPT 2200,VMIN,VMAX
      VSTEP=0.01*(VMAX-VMIN)
      IF(VSTEP.LE.0.0) STOP'VELOCITIES IN WRONG ORDER'
      TYPE 3000
 3000 FORMAT('$START AND END TIME FOR ANALYSIS(MSEC) (REAL)?,? ')
      ACCEPT 2200,TSTART,TEND
      TYPE 1950
 1950 FORMAT('$INITIAL SOURCE/RECEIVER OFFSET AND RECEIVER SPACING',
     +'(M) (REAL)?,? ')
      ACCEPT 2200,XOFF,XSTEP
      TYPE 1600
 1600 FORMAT('$DEFAULT ALL OTHER PARAMETERS (Y/N)? ')
      ACCEPT 1100,IANS
      IF(IANS.EQ.'Y') GOTO 9999
C
C  IF NOT DEFAUTING ALL OTHER PARAMETERS ASK FOLLOWING QUESTIONS
C
      TYPE 1700
 1700 FORMAT(/,'$INTERPOLATION LEVEL FOR I/P DATA(1,2 OR 4)',
     +'(INTEGER)? ')
      ACCEPT 2100,M
      TYPE 1800
 1800 FORMAT(' SCALING OF INTERPOLATED TRACE ? ',/,
     %' LT.0 - NO SCALING',/,
     %' EQ.0 - SCALED TO UNIT ENERGY',/,
     %'$GT.0 - SCALED TO INVERSE ENERGY ')
      ACCEPT 2100,INDEN
      IF(NTRIN.GT.48) GOTO 950
      TYPE 1850
 1850 FORMAT('$MODIFY/INSERT TRACE MUTE FUNCTION (Y/N)?')
      ACCEPT 1100,IANS
      IF(IANS.EQ.'N') GOTO 950
      INMUT=1
      TYPE 1860
 1860 FORMAT(' GIVE MUTE FUNC. IN TRACE ORDER (MSEC)(INTEGER),',
     +' 12 PER LINE')
      ACCEPT 2100,(MUTE(I),I=1,NTRIN)
 950  TYPE 1920
 1920 FORMAT('$GIVE VELOCITY STEP SUCH THAT ((VMAX-VMIN)/VSTEP)<121',
```

```
     +' (KM/SEC)(REAL)? ')
      ACCEPT 2200,VSTEP
      TYPE 3100
 3100 FORMAT('$GIVE LENGTH OF AVERAGING WINDOW (MSEC) (REAL)? ')
      ACCEPT 2200,TGATE
C
C  GET DATA FOR RHVCON PROGRAM
C
      TYPE 3200
 3200 FORMAT('$LENGTH OF TIME AXIS IN INCHES (REAL)?' )
      ACCEPT 2200,XLEN
      TYPE 3300
 3300 FORMAT('$MINIMUM CONTOUR & CONTOUR INTERVAL (REAL)?,?')
      ACCEPT 2200,CMIN,CINT
C
C  RETURN
C
 9999 RETURN
      END




      SUBROUTINE INDKVA(FSPECR)
C
C    INPUT PARAMETER SUBROUTINE FOR
C    VELOCITY ANALYSIS PROGRAM
C
      REAL*8 FSPECR,FSPECI
      INTEGER*2 PARM(50),MUTE(48),IHBLK(256),ICONST(3),
     +SPEC(39),EXT(4)
      LOGICAL*1 FBUF(16),IERR
C
      COMMON/FILE/ IHFLG,IRD,NWDIN,NBLKIN,NVTOT,INDEN
      COMMON/DAT/ NSAMP,NTRIN,NDELAY,M,NSAMP2,NOP,NGATH,NANL
      COMMON/HEAD/ MUTE,IHBLK,ICONST
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/TIME/ TSTART,TEND,TSTEP,TGATE,TINDEL
      COMMON/VEL/ VMIN,VMAX,VSTEP
      COMMON/PASS/ PARM
C
      EQUIVALENCE (PARM(1),XLEN),(PARM(3),CMIN),(PARM(5),CINT)
      EQUIVALENCE (SPEC(16),FSPECI)
C
      DATA EXT/4*0/
C
      CALL GETSTR(1,FBUF,14,IERR)
      IF(IERR.NE..FALSE.) STOP'ILLEGAL FILE NAME'
      IF(ICSI(SPEC,EXT,FBUF,,0).NE.0) STOP'FILE NAME ERROR'
      FSPECR=FSPECI
      READ(1,2100)NTRIN,NSAMP,IHFLG,NGATH,NANL
 2100 FORMAT(1216)
      READ(1,2100) M,INDEN,INMUT
      IF(INMUT.GT.0.AND.NTRIN.GT.48) STOP
     +'NO MUTE ALLOWED IF I/P GATHER HAS > 48 TRACES'
      IF(INMUT.GT.0) READ(1,2100) (MUTE(I),I=1,NTRIN)
      READ(1,2200)VMIN,VMAX,VSTEP
      IF(IHFLG.EQ.0) READ(1,2200) TSTART,TEND,TSTEP,TGATE,TINDEL
      IF(IHFLG.EQ.1) READ(1,2200) TSTART,TEND,TSTEP,TGATE
      READ(1,2200)XOFF,XSTEP
 2200 FORMAT(5F10.0)
      READ(1,2200)XLEN,CMIN,CINT
C
C  RETURN
C
      RETURN
      END
```

```
      SUBROUTINE INCHVA(FSPECR)
C
C  SUBROUTINE TO UNWRAP CHAINED INFORMATION
C  FOR VELOCITY ANALYSIS PROGRAM
C
C  R.W.HOBBS, APRIL 1983
C
      REAL*8 FSPECR,FSPECI
      INTEGER*2 PARM(50)
C
      COMMON/FILE/ IHFLG,IRD,NWDIN,NBLKIN,NVTOT,INDEN
      COMMON/DAT/ NSAMP,NTRIN,NDELAY,M,NSAMP2,NOP,NGATH,NANL
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/TIME/ TSTART,TEND,TSTEP,TGATE,TINDEL
      COMMON/VEL/ VMIN,VMAX,VSTEP
      COMMON/PASS/ PARM
C
      EQUIVALENCE (PARM(7),RTSTRT),(PARM(9),RTEND),
     +(PARM(11),RTGATE),(PARM(13),LGATH),(PARM(14),LANL),
     +(PARM(19),FSPECI),(PARM(23),NTRAS),(PARM(24),NSMPIN),
     +(PARM(25),RTSTEP),(PARM(27),RTNDEL),(PARM(29),IM),
     +(PARM(30),IINDEN),(PARM(31),RVMIN),(PARM(33),RVMAX),
     +(PARM(35),RVSTEP),(PARM(37),RXOFF),(PARM(39),RXSTEP),
     +(PARM(41),LOP)
C
C  PASS PARAMETERS TO LOCAL VARIABLES
C
      FSPECR=FSPECI
      NTRIN=NTRAS
      NSAMP=NSMPIN
      TSTART=RTSTRT
      TEND=RTEND
      TGATE=RTGATE
      TSTEP=RTSTEP
      TINDEL=RTNDEL
      M=IM
      NOP=LOP
      NGATH=LGATH
      NANL=LANL
      INDEN=IINDEN
      VMIN=RVMIN
      VMAX=RVMAX
      VSTEP=RVSTEP
      XOFF=RXOFF
      XSTEP=RXSTEP
      IHFLG=1
C
C  RETURN
C
      RETURN
      END




      SUBROUTINE VELSET
C
C   INPUT PARAMETER CALCULATION SUBROUTINE FOR
C   VELOCITY ANALYSIS PROGRAM
C
      INTEGER*2 MUTE(48),IHBLK(256),ICONST(3),MUTDUM(48),IDEAD(48)
C
      COMMON/SETUP/ NSAMOT,NSTEP,NT0,ISTRT,IAPST,ISMPUT,NGATE,
     +N,IRES,NDEL
      COMMON/FILE/ IHFLG,IRD,NWDIN,NBLKIN,NVTOT,INDEN
      COMMON/DAT/ NSAMP,NTRIN,NDELAY,M,NSAMP2,NOP,NGATH,NANL
      COMMON/HEAD/ MUTE,IHBLK,ICONST
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/TIME/ TSTART,TEND,TSTEP,TGATE,TINDEL
      COMMON/VEL/ VMIN,VMAX,VSTEP
      COMMON/ADR/ IA,IB,IC,ID,IE,IX,IM,IV,IT,IZ
```

```
C
      EQUIVALENCE (IHBLK(55),IDELAY),(IHBLK(58),NSMP),
     +(IHBLK(121),MUTDUM(1)),(IHBLK(169),IDEAD(1))
C
C  READ H/BLK FROM DISK FILE
C
      IF(IREADW(256,IHBLK,0,IRD).LT.0) STOP'H/BLK READ ERR'
C
C  READ H/BLK CONSTANTS
C
      IF(IHFLG.EQ.1) TINDEL=FLOAT(IDELAY)
      IDELAY=IFIX(TINDEL)
      NDELAY=IDELAY
      IF(IHFLG.EQ.1.AND.NSAMP.NE.NSMP)STOP
     +'DATA FILE AND H/BLK HAVE DIFFERENT SAMPLES/TRACE NUMBER'
      IF(M.EQ.0) M=1
      FSAMP=1.0/TSTEP
C
C  SET UP TIME WINDOW FOR TRACE READS
C
      TLAST=TINDEL+(FLOAT(NSAMP)*TSTEP)
      IF(TSTART.EQ.0.0.AND.TEND.EQ.0.0) TSTART=TINDEL
      IF(TEND.EQ.0.0) TEND=TLAST
      TRANGE=TEND-TSTART
      NSAMOT=IFIX(TRANGE*FSAMP)
      IF(NSAMOT.GT.4096) STOP'TIME WINDOW TOO LONG'
      NSTEP=IFIX((TRANGE*FSAMP)/256.0)+1
      NT0=(IFIX(TRANGE*FSAMP))/NSTEP+1
      IF(TGATE.EQ.0.0) TGATE=3*FLOAT(NSTEP)*TSTEP
      ISTRT=1
      IAPST=IFIX((TINDEL-TSTART)*FSAMP)
      IF(IAPST.LT.0) ISTRT=1-IAPST
      IF(IAPST.LT.0) IAPST=0
      ISMPUT=IFIX((AMIN1(TLAST,TEND)-AMAX1(TSTART,TINDEL))*FSAMP)
      IDELAY=IFIX(TSTART)
      IF(NTRIN.GT.48) GOTO 110
      DO 100 I=1,NTRIN
      IF(IHFLG.EQ.0) MUTDUM(I)=-1
      MUTE(I)=MAX0(0,MUTDUM(I)+NDELAY-IDELAY,MUTE(I)-IDELAY)
      IF(MUTE(I).GE.IFIX(TEND)) IDEAD(I)=2
      MUTE(I)=IFIX(FLOAT(MUTE(I))*FSAMP)
 100  IF(MUTE(I).GE.NSAMP) MUTE(I)=NSAMP
C
C  SET UP ALL OTHER CONSTANTS
C
 110  NGATE=2
 125  IF(NGATE.GE.IFIX(TGATE*FSAMP)) GOTO 150
      NGATE=NGATE*2
      GOTO 125
 150  NSAMP2=2
 200  IF(NSAMP2.GE.NSAMOT) GOTO 250
      NSAMP2=NSAMP2*2
      GOTO 200
 250  N=NT0-2
      IRES=1
      NDEL=NSTEP-(NGATE/2)
 390  IF(NDEL.GE.0) GOTO 400
      N=N-2
      IRES=IRES+1
      NDEL=NDEL+NSTEP
      GOTO 390
 400  CONTINUE
C
C  OPTIMIZE AP USAGE
C
      RDAT=FLOAT(3*NSAMOT)
 450  RSUM=RDAT+FLOAT(M*NSAMP2+2+NSAMOT+6)
      IF(RSUM.LE.24576.0) GOTO 500
      IF(M.EQ.1) STOP'I/P TRACES TOO LONG'
      M=M/2
      TYPE 2000,M
 2000 FORMAT(' **WARNING** INTERPOLATION LEVEL REDUCED TO:',I3)
```

```
      GOTO 450
 500  ICONST(1)=IFIX(TSTART*FSAMP)
      ICONST(2)=ICONST(1)*M
      ICONST(3)=NSAMOT*M
      IF(NTRIN.GT.48) GOTO 700
      DO 600 I=1,NTRIN
 600  MUTE(I)=MUTE(I)*M
C
C  SET UP AP ADDRESSES
C
 700  IA=0
      IB=IA+NSAMOT
      IC=IB+NSAMOT
      ID=IC+NSAMOT
      IE=ID+M*NSAMP2+2
      IX=IE+NSAMOT
      IM=IX+1
      IV=IM+1
      IT=IV+1
      IZ=24575
C
C     BLOCKING PARAMETERS
C
      NVTOT=IFIX((VMAX-VMIN)/VSTEP+1.0)
      NBLKIN=NSAMP/128
      NWDIN=NSAMP*2
      RETURN
      END




      SUBROUTINE VELSUB(FSPECR)
C
C  SUBROUTINE USED IN VELOCITY ANALYSIS PACKAGE TO
C  CALCULATE SEMBLANCE
C  WRITTEN BY R.W.HOBBS DECEMBER 1983
C
      VIRTUAL SIES(4096),ISEMB(32767),ISORT(32767)
      REAL*8 FSPECR
      REAL*4 XMVT(5),RSEM(121)
      INTEGER*4 JCDPNO
      INTEGER*2 IHBLK(256),MUTE(48),IDEAD(48),
     $ICONST(3),IT0LYR(10),IVLYR(10)
C
      COMMON/SETUP/ NSAMOT,NSTEP,NT0,ISTRT,IAPST,ISMPUT,NGATE,
     +N,IRES,NDEL
      COMMON/FILE/ IHFLG,IRD,NWDIN,NBLKIN,NVTOT,INDEN
      COMMON/DAT/ NSAMP,NTRIN,NDELAY,M,NSAMP2,NOP,NGATH,NANL
      COMMON/HEAD/ MUTE,IHBLK,ICONST
      COMMON/GEOM/ XOFF,XSTEP,FSAMP
      COMMON/TIME/ TSTART,TEND,TSTEP,TGATE,TINDEL
      COMMON/VEL/ VMIN,VMAX,VSTEP
      COMMON/ADR/ IA,IB,IC,ID,IE,IX,IM,IV,IT,IZ
C
      EQUIVALENCE (XMVT(1),X),(XMVT(2),RM),(XMVT(3),V),(XMVT(4),T),
     +(IHBLK(11),JCDPNO),(IHBLK(100),NLYR),(IHBLK(101),IT0LYR(1)),
     +(IHBLK(111),IVLYR(1)),(IHBLK(169),IDEAD(1))
C
      XMVT(5)=1.0
      CALL JJCVT(JCDPNO)
      CALL IJCVT(JCDPNO,ICDPNO)
C
C  IF SEMBLANCE TO BE STORED OPEN DR: DISK FILE
C
      VSAMP=VMIN*TSTEP
      VSSAMP=VSTEP*TSTEP
      NDEL=NDEL+IA
      IRES=IRES+IC
      IAPST=IAPST+ID
      TSOUT=FLOAT(NSTEP)*TSTEP
```

```
      TGATE=FLOAT(NGATE)*TSTEP
      RM=FLOAT(M)
C
C  WRITE OUT CONSTANTS TO OUTPUT FILE
C
      WRITE(2) NSAMP,NTRIN,NDELAY,M,NGATH
      WRITE(2) XOFF,XSTEP,FSAMP
      WRITE(2) TSTART,TEND,TSOUT,TGATE
      WRITE(2) VSTEP,VMIN,VMAX
      WRITE(2) NT0,ICDPNO,NOP,NLYR,FSPECR
C
C  LOOP FOR EACH INPUT GATHER
C
      DO 980 IG=1,NGATH
      V=VSAMP
      T=FLOAT(NSAMOT+ICONST(1))
      LOC=1
C
C  START VELOCITY LOOP
C
      DO 900 I=1,NVTOT
      CALL APWAIT
      CALL VCLR(0,1,24576)
      JBLKIN=((IG-1)*NTRIN*NBLKIN)+1
      CALL IRDRXM(SIES(1),IRD,NWDIN,JBLKIN,1)
      X=XOFF
      CALL APWR
C
C  LOOP FOR ALL INPUT TRACES
C
      DO 800 J=1,NTRIN
      CALL IWAIT(IRD)
      CALL APPUT(XMVT,IX,5,2)
      CALL VCLR(ID,1,M*NSAMP2+2)
      CALL APWAIT
      CALL APPTXM(SIES(ISTRT),IAPST,ISMPUT,2)
      JBLKIN=JBLKIN+NBLKIN
      CALL APWAIT
      CALL IRDRXM(SIES(1),IRD,NWDIN,JBLKIN,1)
      IF(NTRIN.LE.48) GOTO 600
      IMUTE=-1
      GOTO 750
 600  IMUTE=MUTE(J)
      IF(IHFLG.EQ.1
     +         .AND.IDEAD(J).GE.2) GOTO 800
C
C  NORMALIZE AND DO INTERPOLATION
C
 750  IF(INDEN.LT.0) GOTO 700
      CALL SVESQ(ID,1,IZ,NSAMOT)
      IF(INDEN.EQ.0) CALL VSQRT(IZ,1,IZ,1,1)
      CALL VSDIV(ID,1,IZ,ID,1,NSAMOT)
C
C  TAPER BOTH ENDS OF DATA
C
 700  CALL TAPER(ID,1,ID,1,10)
      CALL TAPER(ID+NSAMOT-1,-1,ID+NSAMOT-1,-1,10)
C
C  INTERPOLATE
C
      CALL RFFT(ID,NSAMP2,1)
      CALL RFFTSC(ID,NSAMP2,3,1)
      CALL RFFTSC(ID,M*NSAMP2,-3,0)
      CALL RFFT(ID,M*NSAMP2,-1)
C
C  APPLY MUTE AND TAPER THEN DO NMO
C
      CALL NMO(IX,IM,IV,IT,ICONST(2),ID,ICONST(3),ICONST(1),IE,NSAMOT,
     $IC,IMUTE)
C
C  ADD AMPLITUDE TO STACK AND AND ENERGY TO SUM
C
```

```
      CALL VADD(IA,1,IE,1,IA,1,NSAMOT)
      CALL VSQ(IE,1,IE,1,NSAMOT)
      CALL VADD(IB,1,IE,1,IB,1,NSAMOT)
 800  X=X+XSTEP
C
C  WAIT FOR ALL DONE AND CALCULATE SEMBLANCE
C
      CALL IWAIT(IRD)
      CALL APWR
      CALL VMUL(IB,1,IC,1,IB,1,NSAMOT)
      CALL LVNOT(IB,1,IC,1,NSAMOT)
      CALL VADD(IB,1,IC,1,IB,1,NSAMOT)
      CALL VSQ(IA,1,IA,1,NSAMOT)
      CALL VDIV(IB,1,IA,1,IA,1,NSAMOT)
C
C  TAKE HANNING AVERAGE
C
      CONST=1000.0/FLOAT(NGATE/2)
      CALL APPUT(CONST,IZ,1,2)
      CALL VCLR(IB,1,(NSAMOT+NT0))
      CALL APHAN(NDEL,ID,IRES,NGATE,NSTEP,N)
      CALL APWD
      CALL VSMUL(IC,1,IZ,IC,1,NT0)
      CALL VFIX(IC,1,IC,1,NT0)
      CALL APWR
      CALL APGTXM(ISEMB(LOC),IC,NT0,1)
      CALL APWD
      LOC=LOC+NT0
 900  V=V+VSSAMP
C
C  DO SORT FROM VELOCITY SLICE TO TIME SLICE ORDER
C
      LOC=1
      DO 920 I=1,NVTOT
      JSTART=I
      DO 910 J=1,NT0
      ISORT(JSTART)=ISEMB(LOC)
      LOC=LOC+1
 910  JSTART=JSTART+NVTOT
 920  CONTINUE
      LOC=1
      CONST=0.001
      CALL APWAIT
      CALL APPUT(CONST,IZ,1,2)
      CALL APWD
C
C  WRITE OUT SEMBLANCE DATA AND POSITION OF MAXIMUM IN EACH TIME SLICE
C
      DO 950 I=1,NT0
      CALL APPTXM(ISORT(LOC),1,NVTOT,1)
      CALL APWD
      CALL VFLT(1,1,1,1,NVTOT)
      CALL VSMUL(1,1,IZ,1,1,NVTOT)
      CALL MAXV(1,1,2048,NVTOT)
      CALL APWR
      CALL APGET(RSEM,1,NVTOT,2)
      CALL APGET(IMAX,2049,1,1)
      CALL APWD
      WRITE(2) NVTOT,IMAX,(RSEM(J),J=1,NVTOT)
 950  LOC=LOC+NVTOT
C
C  LOOP FOR NEXT GATHER
C
 980  CONTINUE
C
C  WRITE OUT VELOCITY FILTER DATA RETURN
C
      IF(NOP.EQ.0) GOTO 960
      DO 955 I=1,NLYR
 955  WRITE(2) IT0LYR(I),IVLYR(I)
 960  RETURN
      END
```

```
C
C VELOCITY ANALYSIS CONTOUR PROGRAM
C THIS TAKES UNFORMATTED INPUT
C FROM OUTPUT OF VELOCITY ANALYSIS
C PROGRAM AND CONTOURS AND ANOTATES IT
C
C    WRITTEN M.J.POULTER
C    MODIFIED R.W.HOBBS APRIL 1983
C                       TO ALLOW CHAINING TO FROM VELANL
C                       TO CHAIN TO PLOTTING ROUTINES
C                       USING SEMPLT
C                             VELTRD (MODIFIED VERSATEC PROG THREAD)
C                             VELRSL (MODIFIED VERSATEC PROG RASL)
C                             VELPLT (DISK TO MATRIX OUTPUT PROG)
C
      VIRTUAL Z(121,256)
      REAL*8 FSPECR,FSPECO
      REAL*4 X(256),Y(121),CZ(20),Z,SEMPLT(2),VECBIN(2),PARBIN(2)
      INTEGER*2 PTR(20),NMAX(256),PARM(40),IT0LYR(10),IVLYR(10)
      LOGICAL*1 SWCHES(5)
      COMMON /PPEP1/ IX1,IY1,IX2,IY2,ISCAN,NSCAN,NBAND,NIPS,NIP0,
     1               NIPM1,LYNES,NIBSX,MSGLVL,XDOTS,YDOTS,PREF(2),
     2               RORG(2),PORT(2,2),IEND(4),ALMT,FACT,JPEN,XOFF,
     3               XFAC,YOFF,YFAC,NBITS,NBITM1,NBYTES,NBYTM1,MSK,LMSK
      EQUIVALENCE (PARM(1),XLEN),(PARM(3),CMIN),(PARM(5),CINT),
     $(PARM(7),TSTART),(PARM(9),TEND),(PARM(11),TSTEP),
     $(PARM(13),TINDEL),(PARM(15),FSPECO),(PARM(19),NTRIN),
     $(PARM(20),NSAMP),(PARM(21),NGATH)
      DATA SWCHES/.TRUE.,.TRUE.,.TRUE.,.TRUE.,.FALSE./
      DATA NC/19/
      DATA SEMPLT(1)/6RDK1SEM/,SEMPLT(2)/6RPLTSAV/
      DATA DKHN/3RDK /,DRHN/3RDR /
      DATA VECBIN/6RDR4VEC,6RTR2BIN/
      DATA PARBIN/6RDR4PAR,6RM  BIN/
C
C  CHECK FOR CHAIN
C
      CALL RCHAIN(ICHFLG,PARM,6)
C
C  DELETE BIN FILES FROM DR5:
C
      IF(IFETCH(DKHN).NE.0) STOP'DK FETCH ERR'
      IF(IFETCH(DRHN).NE.0) STOP'DR FETCH ERR'
      ICHN=IGETC()
      CALL IDELET(ICHN,VECBIN)
      CALL IFREEC(ICHN)
      ICHN=IGETC()
      CALL IDELET(ICHN,PARBIN)
      CALL IFREEC(ICHN)
      IF(ICHFLG.LT.0) GO TO 250
      TYPE 1000
 1000 FORMAT(' GIVE AXIS LENGTH IN INCHES, [RET] FOR DEFAULT ? ',$)
      ACCEPT 2000,XLEN
 2000 FORMAT(6F10.0)
  230 TYPE 1100
 1100 FORMAT(' GIVE MINIMUM CONTOUR & CONTOUR INTERVAL, ',
     $'[RET] FOR DEFAULT ?,? ',$)
      ACCEPT 2000,CMIN,CINT
  250 IF(XLEN.EQ.0.0) XLEN=10.0
      IF(CMIN.EQ.0.0) CMIN=0.2
      IF(CINT.EQ.0.0) CINT=0.1
      NC=IFIX((1.0-CMIN)/CINT)+1
      IF(NC.LE.20) GO TO 220
      TYPE 1210
 1210 FORMAT(' TOO MANY CONTOURS, TRY AGAIN |')
      IF(ICHFLG.LT.0) STOP'CHAINED DATA ERROR'
      GO TO 230
  220 CMAX=CMIN+((NC-1)*CINT)
      CZ(1)=CMAX
      DO 240 I=2,NC
      CZ(I)=CZ(I-1)-CINT
  240 CONTINUE
```

```
C
C READ IN CONTROL DATA
C
      OPEN(UNIT=2,TYPE='OLD',NAME='DR5:CONTUR.DAT',
     +FORM='UNFORMATTED',ERR=900)
      READ(2) LNUM,NCHAN,NDELAY,M,LGATH
      READ(2) XSTART,XSTEPR,FSAMP
      READ(2) T01,T02,T0STEP,TGATE
      READ(2) VSTEP,VMIN,VMAX
      READ(2) NT0,ICDPNO,NOP,NLYR,FSPECR
      TSTART=T01
      TEND=T02
      TSTEP=1.0/FSAMP
      TINDEL=FLOAT(NDELAY)
      FSPECO=FSPECR
      NTRIN=NCHAN
      NSAMP=LNUM
      XSTEP=XLEN/FLOAT(NT0-1)
      NGATH=LGATH
      LYNES=300
      CALL PLOTS(0,0,0)
      PORT(1,1)=5.0*NGATH+8.0
C
C  LOOP FOR EACH GATHER TO BE PLOTTED
C
      DO 200 IG=1,NGATH
      IF(IG.GT.1) CALL PLOT(5.0,0.0,-3)
      IXMP=NT0+1
C
C READ IN THE SEMBLANCE DATA
C
      DO 10 JT0=1,NT0
      IXMP=IXMP-1
      READ(2) NPT,NMAX(JT0),(Z(J,IXMP),J=1,NPT)
      IF(NPT.GT.NPT)NPT=NPT
   10 CONTINUE
      YSTEP=4.0/FLOAT(NPT-1)
C
C SET UP COORDINATES OF CONTOUR ARRAY
C
      Y(1)=0.3
      DO 20 L=2,NPT
   20 Y(L)=Y(L-1)+YSTEP
      X(1)=10.0-XLEN
      DO 30 LL=2,NT0
   30 X(LL)=X(LL-1)+XSTEP
      IXMP=NT0+1
C
C USE CONSYS CONTOUR PACKAGE
C
      CALL CONTUR(Y,NPT,X,NT0,Z,121,CZ,NC,PTR,SWCHES,
     % , , , , )
C
C PLOT TIME GRID
C
      T0LEN=T02-T01
      IF(T0LEN.LT.1000.0) GOTO 35
      TUNIT=1000.0*(X(NT0)-X(1))/T0LEN
      TTUNIT=TUNIT/10.0
      T0BEG=FLOAT(IFIX(T01+1000.0)/1000)
      XOFF=(T0BEG-(T01/1000.0))*TUNIT
      TINIT=1.0
      GOTO 45
   35 TUNIT=100.0*(X(NT0)-X(1))/T0LEN
      TTUNIT=TUNIT/10.0
      T0BEG=FLOAT(IFIX(T01+100.0))
      XOFF=((T0BEG/100.0)-(T01/100.0))*TUNIT
      TINIT=100.0
C
C PLOT SECOND GRID
C
   45 TNUM=T0BEG
```

```
      XPT=X(NT0)-XOFF
   50 CALL PLOT(Y(1),XPT,+3)
      CALL PLOT(Y(NPT),XPT,+2)
      CALL NUMBER(0.0,XPT,0.05,TNUM,0.0,1)
      XPT=XPT-TUNIT
      TNUM=TNUM+TINIT
      IF(XPT.GT.X(1))GOTO 50
C
C PLOT TENTHS OF SECONDS GRID
C
      LMSK="1403
      MSK=+1
      XPT=X(NT0)-XOFF
   60 CALL PLOT(Y(1),XPT,+3)
      CALL PLOT(Y(NPT),XPT,+2)
      XPT=XPT-TTUNIT
      IF(XPT.GT.X(1))GOTO 60
      XPT=X(NT0)-XOFF
   70 CALL PLOT(Y(1),XPT,+3)
      CALL PLOT(Y(NPT),XPT,+2)
      XPT=XPT+TTUNIT
      IF(XPT.LT.X(NT0))GOTO 70
C
C VELOCITY GRID
C
      VLEN=VMAX-VMIN
      VUNIT=(Y(NPT)-Y(1))/VLEN
      VTUNIT=VUNIT/10.0
      VBEG=FLOAT(IFIX(VMIN+1.0))
      YOFF=(VBEG-VMIN)*VUNIT
C
C PLOT KM/S GRID
C
      LMSK=-1
      MSK=0
      YPT=Y(1)+YOFF
      VNUM=VBEG
      CALL SYMBOL(1.9,10.2,0.05,'RMS VELOCITY KM/S',0.0,17)
   80 CALL PLOT(YPT,X(NT0),+3)
      CALL PLOT(YPT,X(1),+2)
      CALL NUMBER(YPT-0.05,X(NT0)+0.02,0.05,VNUM,0.0,1)
      YPT=YPT+VUNIT
      VNUM=VNUM+1.0
      IF(YPT.LT.Y(NPT))GOTO 80
C
C PLOT TENTHS GRID
C
      LMSK="1403
      MSK=-1
      YPT=Y(1)+YOFF
   90 CALL PLOT(YPT,X(NT0),+3)
      CALL PLOT(YPT,X(1),+2)
      YPT=YPT-VTUNIT
      IF(YPT.GT.Y(1))GOTO 90
      YPT=Y(1)+YOFF
  100 CALL PLOT(YPT,X(NT0),+3)
      CALL PLOT(YPT,X(1),+2)
      YPT=YPT+VTUNIT
      IF(YPT.LT.Y(NPT))GOTO 100
C
C PLOT POSITION OF MAX PTS
C
      LMSK=-1
      MSK=0
      CALL SYMBOL(Y(NPT),10.2,0.05,'  PEAK',0.0,6)
      CALL SYMBOL(Y(NPT),10.1,0.05,'SEMBLANCE',0.0,9)
      IXMP=NT0+1
      DO 40 JT0=1,NT0
      IXMP=IXMP-1
      XP=X(IXMP)
      YP=Y(NMAX(JT0))
      CALL SYMBOL(YP,XP,0.02,0,0.0,-1)
```

```
      SLEN=Y(NPT)+0.5*Z(NMAX(JT0),IXMP)
      CALL PLOT(Y(NPT),XP,+3)
      CALL PLOT(SLEN,XP,+2)
   40 CONTINUE
C
C  LOOP FOR NEXT GATHER
C
  200 CONTINUE
C
C DO ANOTATION
C
      PSIZ=0.1
      X9=999.0
      XA=5.0
      YA=10.3
      CALL SYMBOL(XA,YA,PSIZ,'MEAN CMP GATHER NUMBER: ',0.0,24)
      FNUM=ICDPNO
      CALL NUMBER(X9,X9,PSIZ,FNUM,0.0,-1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,
     $'VELOCITY ANALYSIS SEMBLANCE CONTOURS',0.0,36)
      YA=YA-0.3
      CALL SYMBOL(XA,YA,PSIZ,'PROCESSING PARAMETERS',0.0,21)
      CALL PLOT(XA,YA-0.03,+3)
      CALL PLOT(XA+2.1,YA-0.03,+2)
      YA=YA-0.3
      CALL SYMBOL(XA,YA,PSIZ,'NO. OF CHANNELS = ',0.0,18)
      FNUM=NCHAN
      CALL NUMBER(X9,X9,PSIZ,FNUM,0.0,-1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'SAMPLES PER CHANNEL = ',0.0,22)
      FNUM=LNUM
      CALL NUMBER(X9,X9,PSIZ,FNUM,0.0,-1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'TRACE DELAY MS = ',0.0,17)
      FNUM=NDELAY
      CALL NUMBER(X9,X9,PSIZ,FNUM,0.0,1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'LEVEL OF INTERPOLATION = ',0.0,25)
      FNUM=M
      CALL NUMBER(X9,X9,PSIZ,FNUM,0.0,-1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'CHANNEL 1 OFFSET M = ',0.0,21)
      CALL NUMBER(X9,X9,PSIZ,XSTART,0.0,1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'CHANNEL SPACING M = ',0.0,20)
      CALL NUMBER(X9,X9,PSIZ,XSTEPR,0.0,1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'SAMPLING INTERVAL MS = ',0.0,23)
      CALL NUMBER(X9,X9,PSIZ,TSTEP,0.0,1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'START OF ANALYSIS MS = ',0.0,23)
      CALL NUMBER(X9,X9,PSIZ,T01,0.0,-1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'END OF ANALYSIS MS = ',0.0,21)
      CALL NUMBER(X9,X9,PSIZ,T02,0.0,-1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'TIME STEP MS = ',0.0,15)
      CALL NUMBER(X9,X9,PSIZ,T0STEP,PSIZ,1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'HANNING GATEWIDTH MS = ',0.0,23)
      CALL NUMBER(X9,X9,PSIZ,TGATE,0.0,-1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'START VELOCITY KM/S = ',0.0,22)
      CALL NUMBER(X9,X9,PSIZ,VMIN,0.0,2)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'END VELOCITY KM/S = ',0.0,20)
      CALL NUMBER(X9,X9,PSIZ,VMAX,0.0,2)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'VELOCITY STEP KM/S = ',0.0,21)
      CALL NUMBER(X9,X9,PSIZ,VSTEP,0.0,2)
      YA=YA-0.2
```

```
      CALL SYMBOL(XA,YA,PSIZ,'MIN. CONTOUR VALUE = ',0.0,21)
      CALL NUMBER(X9,X9,PSIZ,CMIN,0.0,2)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'CONTOUR INTERVAL = ',0.0,19)
      CALL NUMBER(X9,X9,PSIZ,CINT,0.0,2)
C
C  VELOCITY FILTER ANOTATION
C
      IF(NOP.EQ.0) GOTO 800
      FNUM=NOP
      YA=YA-0.3
      CALL SYMBOL(XA,YA,PSIZ,'VELOCITY FILTERING APPLIED',0.0,26)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'OPERATOR DIMENSION = ',0.0,21)
      CALL NUMBER(X9,X9,PSIZ,FNUM,0.0,-1)
      DO 700 I=1,NLYR
 700  READ(2) IT0LYR(I),IVLYR(I)
      IVDIF=IVLYR(NLYR)-IVLYR(NLYR-1)
      ITDIF=IT0LYR(NLYR)-IT0LYR(NLYR-1)
      ITREM=IFIX(T02)-IT0LYR(NLYR-1)
      IVLYR(NLYR)=IVLYR(NLYR-1)+IFIX(FLOAT(IVDIF)*FLOAT(ITREM)/
     %FLOAT(ITDIF))
      IT0LYR(NLYR)=IFIX(T02)
      VS=250.0*VLEN
      TS=T0LEN/XLEN
      CALL PLOT(-(NGATH-1)*5.0,0.0,-3)
      DO 780 IG=1,NGATH
      IF(IG.GT.1) CALL PLOT(5.0,0.0,-3)
      YV=0.3+(FLOAT(IVLYR(1))-(1000.0*VMIN))/VS
      XV=10.0
      CALL PLOT(YV,XV,+3)
      DO 750 I=1,NLYR
      YV=0.3+(FLOAT(IVLYR(I))-(1000.0*VMIN))/VS
      XV=10.0-(FLOAT(IT0LYR(I))-T01)/TS
 750  CALL PLOT(YV,XV,+2)
 780  CONTINUE
 800  CALL PLOT(0.0,0.0,999)
C
C  CHAIN TO GATHER PLOT PROGRAM SEMPLT
C
      CLOSE(UNIT=2)
C
      CALL RESET
      TYPE 9000
 9000 FORMAT(' VELOCITY CONTOURING SUCCESSFULLY COMPLETED',/)
      CALL CHAIN(SEMPLT,PARM,40)
 900  STOP'OPEN ERROR ON DR5:CONTUR.DAT'
      END
```

```
C
C VELOCITY ANALYSIS STATICS PROGRAM
C THIS TAKES UNFORMATTED INPUT
C FROM OUTPUT OF VELOCITY ANALYSIS
C PROGRAM PERFORMS STATISTICAL ANALYSIS AND ANOTATES IT
C
C    WRITTEN R.W.HOBBS APRIL 1985
C                       TO ALLOW CHAINING TO FROM VELANL
C                       TO CHAIN TO PLOTTING ROUTINES
C                       USING SEMPLT
C                             VELTRD (MODIFIED VERSATEC PROG THREAD)
C                             VELRSL (MODIFIED VERSATEC PROG RASL)
C                             VELPLT (DISK TO MATRIX OUTPUT PROG)
C
      VIRTUAL Z(121),PICK(3,20,256),ZSUM(121,256)
      REAL*8 FSPECR,FSPECO
      REAL*4 X(256),Y(121),SEMPLT(2),VECBIN(2),PARBIN(2)
      INTEGER*2 PARM(40),IT0LYR(10),IVLYR(10)
C
      COMMON /PPEP1/ IX1,IY1,IX2,IY2,ISCAN,NSCAN,NBAND,NIPS,NIP0,
     1               NIPM1,LYNES,NIBSX,MSGLVL,XDOTS,YDOTS,PREF(2),
     2               RORG(2),PORT(2,2),IEND(4),ALMT,FACT,JPEN,XOFF,
     3               XFAC,YOFF,YFAC,NBITS,NBITM1,NBYTES,NBYTM1,MSK,LMSK
C
      EQUIVALENCE (PARM(1),XLEN),(PARM(3),CMIN),(PARM(5),CINT),
     $(PARM(7),LANL)
      EQUIVALENCE (PARM(7),TSTART),(PARM(9),TEND),(PARM(11),TSTEP),
     $(PARM(13),TINDEL),(PARM(15),FSPECO),(PARM(19),NTRIN),
     $(PARM(20),NSAMP),(PARM(21),NGATH)
C
      DATA IA/1/,IB/1024/,IC/2048/
      DATA SEMPLT(1)/6RDK1SEM/,SEMPLT(2)/6RPLTSAV/
      DATA APHN/3RAP /,DRHN/3RDR /
      DATA VECBIN/6RDR4VEC,6RTR2BIN/
      DATA PARBIN/6RDR4PAR,6RM  BIN/
C
C  CHECK FOR CHAIN
C
      CALL RCHAIN(ICHFLG,PARM,7)
C
C  DELETE BIN FILES FROM DR5:
C
      IF(IFETCH(APHN).NE.0) STOP'AP FETCH ERR'
      IF(IFETCH(DRHN).NE.0) STOP'DR FETCH ERR'
      ICHN=IGETC()
      CALL IDELET(ICHN,VECBIN)
      CALL IFREEC(ICHN)
      ICHN=IGETC()
      CALL IDELET(ICHN,PARBIN)
      CALL IFREEC(ICHN)
      IF(ICHFLG.LT.0) GO TO 10
      TYPE 1000
 1000 FORMAT(' GIVE AXIS LENGTH IN INCHES, [RET] FOR DEFAULT ? ',$)
      ACCEPT 2000,XLEN
 2000 FORMAT(6F10.0)
      TYPE 1100
 1100 FORMAT(' GIVE MINIMUM SEMBLANCE LEVEL, [RET] FOR DEFAULT ? ',$)
      ACCEPT 2000,CMIN
      TYPE 1200
 1200 FORMAT(' OUTPUT FORMAT 1 = PICKED MAXIMA SCATTER STATISTICS',/,
     $       '$              2 = SUMMED SEMBLANCE STATISTICS ? ')
      ACCEPT 2100,LANL
 2100 FORMAT(I6)
      IF(LANL.EQ.0) LANL=1
 10   IF(XLEN.EQ.0.0) XLEN=10.0
      IF(CMIN.EQ.0.0) CMIN=0.2
      NANL=LANL
C
C READ IN CONTROL DATA
C
      OPEN(UNIT=2,TYPE='OLD',NAME='DR5:CONTUR.DAT',
     +FORM='UNFORMATTED',ERR=900)
```

```
      READ(2) LNUM,NCHAN,NDELAY,M,LGATH
      READ(2) XSTART,XSTEPR,FSAMP
      READ(2) T01,T02,T0STEP,TGATE
      READ(2) VSTEP,VMIN,VMAX
      READ(2) NT0,ICDPNO,NOP,NLYR,FSPECR
      TSTART=T01
      TEND=T02
      TSTEP=1.0/FSAMP
      TINDEL=FLOAT(NDELAY)
      FSPECO=FSPECR
      NTRIN=NCHAN
      NSAMP=LNUM
      XSTEP=XLEN/FLOAT(NT0-1)
      NGATH=LGATH
C
C  GET AP AND CLEAR OUT VIRTUAL ARRAYS
C
      CALL APINIT(0,0,IAP)
      CALL APGTXM(Z(1),0,121,2)
      CALL APGTXM(PICK(1,1,1),0,15360,2)
      CALL APGTXM(ZSUM(1,1),0,15488,2)
      CALL APGTXM(ZSUM(1,129),0,15488,2)
      CALL APWD
C
C  IF SUMNATION READ, NORMALIZE AND SUM INPUT SEMBLANCE DATA
C
      IF(NANL.EQ.1) GOTO 16
      CONST=1.0/FLOAT(NGATH)
      CALL APPUT(CONST,IC,1,2)
      CALL APWD
      DO 12 IG=1,NGATH
      DO 14 JT0=1,NT0
      READ(2) NPT,NMAX,(Z(I),I=1,NPT)
      CALL APPTXM(ZSUM(1,JT0),IA,NPT,2)
      CALL APPTXM(Z(1),IB,NPT,2)
      CALL APWD
      CALL VSMUL(IB,1,IC,IB,1,NPT)
      CALL VADD(IA,1,IB,1,IA,1,NPT)
      CALL APWR
      CALL APGTXM(ZSUM(1,JT0),IA,NPT,2)
      CALL APWD
 14   CONTINUE
 12   CONTINUE
C
C  INDENTIFY AND STORE MAXIMA AND HALF WIDTHS
C
 16   NTIME=NGATH
      IF(NANL.NE.1) NTIME=1
      DO 29 IG=1,NTIME
      DO 28 JT0=1,NT0
      IF(NANL.EQ.1) GOTO 21
      CALL APPTXM(ZSUM(1,JT0),IA,NPT,2)
      CALL APWD
      CALL MAXV(IA,1,IC,NPT)
      CALL APWR
      CALL APGET(NMAX,IC+1,1,1)
      CALL APGTXM(Z(1),IA,NPT,2)
      CALL APWD
      GOTO 22
C
C  READ DATA FROM FILE
C
 21   READ(2) NPT,NMAX,(Z(I),I=1,NPT)
C
C  COME HERE FOR BOTH OPTIONS
C
 22   IF(Z(NMAX).LT.CMIN) GOTO 28
      PICK(1,IG,JT0)=Z(NMAX)
      PICK(2,IG,JT0)=FLOAT(NMAX)
      SHMAX=0.5*PICK(1,IG,JT0)
      DO 23 I=NMAX,NPT
 23   IF(Z(I).LE.SHMAX) GOTO 24
```

```
 24    NUH=I-NMAX
       DO 25 I=NMAX,1,-1
 25    IF(Z(I).LE.SHMAX) GOTO 26
 26    NLH=NMAX-I
       PICK(3,IG,JT0)=NUH+NLH
 28    CONTINUE
 29    CONTINUE
C
C  INITAILIZE PLOT ROUTINE
C
       CALL APRLSE
       LYNES=300
       CALL PLOTS(0,0,0)
       PORT(1,1)=20.0
C
C SET UP COORDINATES OF CONTOUR ARRAY
C
       YSTEP=4.0/FLOAT(NPT-1)
       Y(1)=0.3
       DO 30 I=2,NPT
 30    Y(I)=Y(I-1)+YSTEP
       X(1)=10.0-XLEN
       DO 35 I=2,NT0
 35    X(I)=X(I-1)+XSTEP
C
C  PLOT BOX
C
       CALL PLOT(Y(1),X(1),3)
       CALL PLOT(Y(NPT),X(1),2)
       CALL PLOT(Y(NPT),X(NT0),2)
       CALL PLOT(Y(1),X(NT0),2)
       CALL PLOT(Y(1),X(1),2)
C
C PLOT TIME GRID
C
       T0LEN=T02-T01
       IF(T0LEN.LT.1000.0) GOTO 40
       TUNIT=1000.0*(X(NT0)-X(1))/T0LEN
       TTUNIT=TUNIT/10.0
       T0BEG=FLOAT(IFIX(T01+1000.0)/1000)
       XOFF=(T0BEG-(T01/1000.0))*TUNIT
       TINIT=1.0
       GOTO 45
 40    TUNIT=100.0*(X(NT0)-X(1))/T0LEN
       TTUNIT=TUNIT/10.0
       T0BEG=FLOAT(IFIX(T01+100.0))
       XOFF=((T0BEG/100.0)-(T01/100.0))*TUNIT
       TINIT=100.0
C
C PLOT SECOND GRID
C
 45    TNUM=T0BEG
       XPT=X(NT0)-XOFF
 50    CALL PLOT(Y(1),XPT,+3)
       CALL PLOT(Y(NPT),XPT,+2)
       CALL NUMBER(0.0,XPT,0.05,TNUM,0.0,1)
       XPT=XPT-TUNIT
       TNUM=TNUM+TINIT
       IF(XPT.GT.X(1))GOTO 50
C
C VELOCITY GRID
C
       VLEN=VMAX-VMIN
       VUNIT=(Y(NPT)-Y(1))/VLEN
       VTUNIT=VUNIT/10.0
       VBEG=FLOAT(IFIX(VMIN+1.0))
       YOFF=(VBEG-VMIN)*VUNIT
C
C PLOT KM/S GRID
C
       YPT=Y(1)+YOFF
       VNUM=VBEG
```

```
      CALL SYMBOL(1.9,10.2,0.05,'RMS VELOCITY KM/S',0.0,17)
 60   CALL PLOT(YPT,X(NT0),+3)
      CALL PLOT(YPT,X(1),+2)
      CALL NUMBER(YPT-0.05,X(NT0)+0.02,0.05,VNUM,0.0,1)
      YPT=YPT+VUNIT
      VNUM=VNUM+1.0
      IF(YPT.LT.Y(NPT))GOTO 60
C
C PLOT POSITION OF MAX PTS
C
      DO 200 IG=1,NTIME
      IXMP=NT0+1
      DO 150 JT0=1,NT0
      IXMP=IXMP-1
      IF(PICK(1,IG,JT0).EQ.0) GOTO 150
      XP=X(IXMP)
      YP=Y(IFIX(PICK(2,IG,JT0)))
      CALL SYMBOL(YP,XP,0.01,0,0.0,-1)
 150  CONTINUE
C
C  LOOP FOR NEXT GATHER
C
  200 CONTINUE
C
C  DO STATISTICS
C
      CALL SYMBOL(Y(NPT),10.2,0.05,'  MEAN',0.0,6)
      CALL SYMBOL(Y(NPT),10.1,0.05,'SEMBLANCE',0.0,9)
      CALL SYMBOL(Y(NPT)+0.7,10.2,0.05,'STANDARD',0.0,8)
      CALL SYMBOL(Y(NPT)+0.7,10.1,0.05,' ERROR',0.0,6)
      SMAX=0.0
      IXMP=NT0+1
      DO 250 JT0=1,NT0
      IXMP=IXMP-1
      AM=0.0
      WT=0.0
      DO 220 IG=1,NTIME
      AM=AM+PICK(1,IG,JT0)*PICK(2,IG,JT0)
 220  WT=WT+PICK(1,IG,JT0)
      IF(WT.EQ.0.0) GOTO 250
      AM=AM/WT
      VR=0.0
      DO 230 IG=1,NTIME
 230  VR=VR+PICK(1,IG,JT0)*((PICK(2,IG,JT0)-AM)**2)
      SE=SQRT(VR/(WT*AMAX0(1,NTIME-1)))
      XP=X(IXMP)
      YP=(AM-1.0)*YSTEP+Y(1)
      CALL SYMBOL(YP,XP,0.05,3,0.0,-1)
      SLEN=Y(NPT)+0.5*WT/FLOAT(NTIME)
      CALL PLOT(Y(NPT),XP,3)
      CALL PLOT(SLEN,XP,2)
      SEL=SE*YSTEP
      CALL PLOT(YP-SEL,XP,3)
      CALL PLOT(YP+SEL,XP,2)
      YP=Y(NPT)+0.7
      CALL PLOT(YP,XP,3)
      CALL PLOT(YP+SEL,XP,2)
      IF(SEL.GT.SMAX) SMAX=SEL
 250  CONTINUE
C
C  PLOT ASPECT RATIOS
C
      YP=Y(NPT)+SMAX+1.0
      IF(YP.LT.5.5) YP=5.5
      SMAX=0.0
      CALL SYMBOL(YP,10.2,0.05,'MEAN ASPECT',0.0,11)
      CALL SYMBOL(YP,10.1,0.05,'   RATIO',0.0,8)
      IXMP=NT0+1
      DO 260 JT0=1,NT0
      IXMP=IXMP-1
      XP=X(IXMP)
      WT=0.0
```

```
      HW=0.0
      DO 270 IG=1,NTIME
      IF(PICK(3,IG,JT0).EQ.0.0) GOTO 270
      WT=WT+PICK(1,IG,JT0)
      HW=HW+PICK(3,IG,JT0)
 270  CONTINUE
      IF(HW.EQ.0.0) GOTO 260
      SEL=WT/(HW*3.0*YSTEP)
      CALL PLOT(YP,XP,3)
      CALL PLOT(YP+SEL,XP,2)
      IF(SEL.GT.SMAX) SMAX=SEL
 260  CONTINUE
C
C  DO ANNOTATION
C
      X9=999.0
      IF(SMAX.LT.0.6) SMAX=0.6
      XA=YP+SMAX+0.3
      YA=10.3
      PSIZ=0.1
      CALL SYMBOL(XA,YA,PSIZ,'MEAN CMP GATHER NUMBER: ',0.0,24)
      FNUM=ICDPNO
      CALL NUMBER(X9,X9,PSIZ,FNUM,0.0,-1)
      YA=YA-0.2
      IF(NANL.EQ.1) CALL SYMBOL(XA,YA,PSIZ,
     $'VELOCITY ANALYSIS STATISTICS PLOT',0.0,33)
      IF(NANL.EQ.2) CALL SYMBOL(XA,YA,PSIZ,
     $'POSITION OF SUMMED SEMBLANCE MAXIMA',0.0,35)
      YA=YA-0.3
      CALL SYMBOL(XA,YA,PSIZ,'PROCESSING PARAMETERS',0.0,21)
      CALL PLOT(XA,YA-0.03,+3)
      CALL PLOT(XA+2.1,YA-0.03,+2)
      YA=YA-0.3
      CALL SYMBOL(XA,YA,PSIZ,'NO. OF CHANNELS = ',0.0,18)
      FNUM=NCHAN
      CALL NUMBER(X9,X9,PSIZ,FNUM,0.0,-1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'SAMPLES PER CHANNEL = ',0.0,22)
      FNUM=LNUM
      CALL NUMBER(X9,X9,PSIZ,FNUM,0.0,-1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'TRACE DELAY MS = ',0.0,17)
      FNUM=NDELAY
      CALL NUMBER(X9,X9,PSIZ,FNUM,0.0,1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'LEVEL OF INTERPOLATION = ',0.0,25)
      FNUM=M
      CALL NUMBER(X9,X9,PSIZ,FNUM,0.0,-1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'CHANNEL 1 OFFSET M = ',0.0,21)
      CALL NUMBER(X9,X9,PSIZ,XSTART,0.0,1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'CHANNEL SPACING M = ',0.0,20)
      CALL NUMBER(X9,X9,PSIZ,XSTEPR,0.0,1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'SAMPLING INTERVAL MS = ',0.0,23)
      CALL NUMBER(X9,X9,PSIZ,TSTEP,0.0,1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'START OF ANALYSIS MS = ',0.0,23)
      CALL NUMBER(X9,X9,PSIZ,T01,0.0,-1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'END OF ANALYSIS MS = ',0.0,21)
      CALL NUMBER(X9,X9,PSIZ,T02,0.0,-1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'TIME STEP MS = ',0.0,15)
      CALL NUMBER(X9,X9,PSIZ,T0STEP,PSIZ,1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'HANNING GATEWIDTH MS = ',0.0,23)
      CALL NUMBER(X9,X9,PSIZ,TGATE,0.0,-1)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'START VELOCITY KM/S = ',0.0,22)
      CALL NUMBER(X9,X9,PSIZ,VMIN,0.0,2)
```

```
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'END VELOCITY KM/S = ',0.0,20)
      CALL NUMBER(X9,X9,PSIZ,VMAX,0.0,2)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'VELOCITY STEP KM/S = ',0.0,21)
      CALL NUMBER(X9,X9,PSIZ,VSTEP,0.0,2)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'MIN. SEMBLANCE VALUE = ',0.0,23)
      CALL NUMBER(X9,X9,PSIZ,CMIN,0.0,2)
      YA=YA-0.3
      IF(NANL.EQ.1) CALL SYMBOL(XA,YA+0.05,0.01,0,0.0,-1)
      IF(NANL.EQ.2) CALL SYMBOL(XA,YA+0.03,0.05,3,0.0,-1)
      CALL SYMBOL(XA+0.1,YA,PSIZ,' POS. OF SEMB. MAXIMA',0.0,21)
      IF(NANL.EQ.2) GOTO 500
      YA=YA-0.2
      CALL SYMBOL(XA,YA+0.03,0.05,3,0.0,-1)
      CALL SYMBOL(XA+0.1,YA,PSIZ,' POS. OF MEAN SEMBLANCE',0.0,23)
C
C  VELOCITY FILTER ANOTATION
C
 500  IF(NOP.EQ.0) GOTO 800
      FNUM=NOP
      YA=YA-0.3
      CALL SYMBOL(XA,YA,PSIZ,'VELOCITY FILTERING APPLIED',0.0,26)
      YA=YA-0.2
      CALL SYMBOL(XA,YA,PSIZ,'OPERATOR DIMENSION = ',0.0,21)
      CALL NUMBER(X9,X9,PSIZ,FNUM,0.0,-1)
      DO 700 I=1,NLYR
 700  READ(2) IT0LYR(I),IVLYR(I)
      IVDIF=IVLYR(NLYR)-IVLYR(NLYR-1)
      ITDIF=IT0LYR(NLYR)-IT0LYR(NLYR-1)
      ITREM=IFIX(T02)-IT0LYR(NLYR-1)
      IVLYR(NLYR)=IVLYR(NLYR-1)+IFIX(FLOAT(IVDIF)*FLOAT(ITREM)/
     %FLOAT(ITDIF))
      IT0LYR(NLYR)=IFIX(T02)
      VS=250.0*VLEN
      TS=T0LEN/XLEN
      YV=0.3+(FLOAT(IVLYR(1))-(1000.0*VMIN))/VS
      XV=10.0
      CALL PLOT(YV,XV,+3)
      DO 750 I=1,NLYR
      YV=0.3+(FLOAT(IVLYR(I))-(1000.0*VMIN))/VS
      XV=10.0-(FLOAT(IT0LYR(I))-T01)/TS
 750  CALL PLOT(YV,XV,+2)
 800  CALL PLOT(0.0,0.0,999)
C
C  CHAIN TO GATHER PLOT PROGRAM SEMPLT
C
      CLOSE(UNIT=2)
C
      CALL RESET
      TYPE 9000
 9000 FORMAT(' VELOCITY STATISTICS SUCCESSFULLY COMPLETED',/)
      CALL CHAIN(SEMPLT,PARM,40)
 900  STOP'OPEN ERROR ON DR5:CONTUR.DAT'
      END
```

```
C
C  THIS IS A FAST PLOT OPTION FOR CDP GATHERS USED
C  FOR VELOCITY ANALYSIS
C
C  R.W.HOBBS 1984
C
      REAL*8 FSPECR,FSPECW
      REAL*4 VELTRD(2),BUF(4096),CONST(3),PAR(3),XMVT(7)
      INTEGER*2 IPBUF(1024),PARM(40),HBLK(256)
      LOGICAL*1 NAME(12)
C
      EQUIVALENCE (PAR(1),XMVT(5))
      EQUIVALENCE (PARM(1),XLEN),(PARM(7),TSTART),
     +(PARM(9),TEND),(PARM(11),TSTEP),(PARM(13),TINDEL),
     $(PARM(15),FSPECR),(PARM(19),NTRIN),(PARM(20),ISAMP),
     $(PARM(21),NGATH)
C
      DATA DRHN/3RDR /,APHN/3RAP /
      DATA XMVT(1)/0.0/,XMVT(3)/1.0/
      DATA FSPECW/12RDR4GPLOT DAT/
      DATA VELTRD(1)/6RDK1VEL/,VELTRD(2)/6RTRDSAV/
C
C  CHECK FOR CHAIN CALL
C
      CALL RCHAIN(ICHFLG,PARM,40)
      IF(ICHFLG.EQ.0) STOP'SEMPLT IS USED BY VELOCITY ANALYSIS ONLY'
C
C  SET UP I/O
C
      IF(IFETCH(DRHN).NE.0) STOP'DR FETCH ERR'
      IF(IFETCH(APHN).NE.0) STOP'AP FETCH ERR'
      CALL APINIT(0,0,IAP)
      IRD=IGETC()
      IWT=IGETC()
      IF(LOOKUP(IRD,FSPECR).LT.0) STOP'NO INPUT FILE'
C
C  SET UP CONSTANTS
C
      IF(IREADW(256,HBLK,0,IRD).LT.0) STOP'H/BLK READ ERR'
      FSAMP=1.0/TSTEP
      JSAMP=ISAMP
      NBLKR=ISAMP/128
      NWDR=2*ISAMP
      NSAMP=IFIX(200.0*XLEN)
      NBLKOT=(NTRIN+4)*4*NGATH+1
      NBLKW=4
      NWDW=1024
      PAR(1)=1.0
      PAR(2)=1.0
      PAR(3)=4.0
      XMVT(2)=((TEND-TSTART)*FSAMP)/(200.0*XLEN)
      TBEG=AMIN1(TSTART,TINDEL)
      NSTART=IFIX((TSTART-TBEG)*FSAMP/XMVT(2))
      JSTART=IFIX((TINDEL-TBEG)*FSAMP)
      XMVT(4)=FLOAT(MAX0(2048,NSAMP+NSTART,JSAMP+JSTART))
      LSAMP=MAX0(JSAMP,NSAMP)
      IA=0
      IB=IA+LSAMP
      IC=IB+LSAMP
      ID=IC+LSAMP
C
C  OPEN O/P FILE
C
      IF(IENTER(IWT,FSPECW,NBLKOT).LT.0) STOP'NO ROOM FOR O/P FILE'
C
C  PUT CONSTANTS IN AP
C
      CALL APPUT(XMVT,24569,7,2)
      CALL APWD
      CALL VRAMP(24573,24574,24001,1,8)
C
C  LOOP FOR EACH GATHER TO BE PLOTTED SO THAT FIRST GATHER APPEARS ON LHS
```

```
C
      IBLKW=NBLKOT-NBLKW
      DO 900 IG=1,NGATH
      IBLKR=(NGATH-IG)*NTRIN*NBLKR+1
C
C  OUTPUT 4 EMPTY TRACES AS SPACE BETWEEN GATHER PLOT AND SEMBLANCE PLOT
C
      CALL VCLR(0,1,1024)
      CALL APWR
      CALL APGET(IPBUF(1),0,1024,1)
      CALL APWD
      DO 200 I=1,4
      IF(IWRITW(NWDW,IPBUF,IBLKW,IWT).LT.0) STOP'PLOT PAD WRITE ERR'
      IBLKW=IBLKW-NBLKW
 200  CONTINUE
C
C  PLOT EACH TRACE
C
      DO 500 I=1,NTRIN
      IF(IREADW(NWDR,BUF,IBLKR,IRD).LT.0) STOP'TRACE READ ERR'
      IBLKR=IBLKR+NBLKR
      CALL VCLR(IA,1,16384)
      CALL APWAIT
      CALL APPUT(BUF,IA,JSAMP,2)
      CALL APWD
C
C  APPLY AGC
C
      CALL VSQ(IA,1,IB,1,JSAMP)
      CALL VRNAVG(IB,IC,JSAMP,50)
      CALL VFILL(IC+50,IC,1,50)
      CALL VFILL(ID-51,ID-51,1,50)
      CALL VSQRT(IC,1,IC,1,JSAMP)
      CALL LVNOT(IC,1,IB,1,JSAMP)
      CALL VADD(IB,1,IC,1,IC,1,JSAMP)
      CALL VDIV(IC,1,IA,1,IA,1,JSAMP)
      CALL VSMUL(IA,1,24575,IA,1,JSAMP)
C
C  RE-SCALE EACH TRACE
C
      CALL VCLR(IC,1,2000)
      CALL NMO(24569,24570,24571,24572,JSTART,IA,JSAMP,
     +NSTART,IC,NSAMP,-1,0)
      CALL VMOV(IC+1999,-1,IA,1,2000)
      CALL APWR
      K=1
      DO 400 J=1,8
      CALL VCLR(2000,1,14384)
      CALL LVGT(0,1,24000+J,0,4096,1,2000)
      CALL RASTER(4096,8192,2000)
      CALL APWR
      CALL APGET(IPBUF(K),8192,128,1)
      CALL APWAIT
 400  K=K+128
C
      IF(IWRITW(NWDW,IPBUF,IBLKW,IWT).LT.0) STOP'PLOT WRITE ERR'
      IBLKW=IBLKW-NBLKW
 500  CONTINUE
C
C  LOOP OR NEXT GATHER
C
 900  CONTINUE
C
      TYPE 9000
 9000 FORMAT(' GATHER PLOT SUCCESSFULLY OUTPUT TO DR4',/)
      CALL CLOSEC(IRD)
      CALL CLOSEC(IWT)
      CALL RESET
      CALL CHAIN(VELTRD,,0)
      CALL EXIT
      END
```

```
C
C  THIS PROGRAM TAKES THE SPOOLED OUTPUT DISC FILE FROM
C  VELRSL AND PUTS IT OUT TO THE PLOTTER
C
C  WRITTEN BY R. W. HOBBS APRIL 1983
C
      DIMENSION IBUF(132,8),IBUFC(1056),INBUF(2000),FBUF(3)
      REAL*8 FSPECR,FSPECP
      INTEGER*2 PARM(4)
      EQUIVALENCE(IBUF(1),IBUFC(1))
      DATA DRHN/3RDR /
      DATA FSPECR/12RDR4VPLOT DAT/,FSPECP/12RDR4GPLOT DAT/
C
C  CHECK FOR CHAIN CALL
C
      CALL RCHAIN(ICHFLG,PARM,4)
C
C  OPEN INPUT FILE
C
      IF(IFETCH(DRHN).NE.0) STOP'DR FETCH ERR'
      IDCH=12
      CALL MTXSET
      CALL MTX(IBUF,0,1)
C
C  SEE IF GATHER PLOT PRESENT
C
      ILK=LOOKUP(IDCH,FSPECP)
      IF(ILK.LT.0) GOTO 50
      IBLK=1
C
C  COPY FILE TO PRINTER
C
 10   IN=IREADW(1024,INBUF,IBLK,IDCH)
      IBLK=IBLK+4
      IF(IN.EQ.1024)GOTO 20
      IF(IN.LT.-1)STOP'READ ERR'
      IF(IN.LE.0)GOTO 40
 20   IPOS=1
      IROW=1
      NROW=IN/128
      IF((IN-(NROW*128)).NE.0)NROW=NROW+1
      NWORD=NROW*132
      CALL MWAIT
      DO 25 I=1,1056
 25   IBUFC(I)=0
      DO 30 I=1,IN
      IBUF(IPOS,IROW)=INBUF(I)
      IPOS=IPOS+1
      IF(IPOS.LE.128)GOTO 30
      IPOS=1
      IROW=IROW+1
 30   CONTINUE
      CALL MTX(IBUF,NWORD,2)
      IF(IN.EQ.1024)GOTO 10
      CALL MWAIT
 40   CONTINUE
      CALL MWAIT
 50   CALL CLOSEC(IDCH)
C
C  LOOKUP VECTOR PLOT FILE
C
      IF(LOOKUP(IDCH,FSPECR).LT.0)STOP'LOOKUP ERR'
      IBLK=1
      IREM=0
      IS=1
C
C  COPY FILE TO PRINTER
C
 60   IN=IREADW(1024,INBUF(IS),IBLK,IDCH)
      IBLK=IBLK+4
      IF(IN.EQ.1024) GOTO 70
      IF(IN.LT.-1) STOP'READ ERR'
```

```
      IF(IN.LE.0) GOTO 100
70    IPOS=1
      IROW=1
      IN=IN+IREM
      NROW=IN/132
      IREM=IN-(NROW*132)
      NWORD=NROW*132
      CALL MWAIT
      DO 75 I=1,1056
75    IBUFC(I)=0
      DO 80 I=1,NWORD
      IBUF(IPOS,IROW)=INBUF(I)
      IPOS=IPOS+1
      IF(IPOS.LE.132) GOTO 80
      IPOS=1
      IROW=IROW+1
80    CONTINUE
      CALL MTX(IBUF,NWORD,2)
      IF(IREM.EQ.0.AND.IN.LE.1024) GOTO 100
      DO 90 I=1,IREM
90    INBUF(I)=INBUF(NWORD+I)
      IS=IREM+1
      GOTO 60
      CALL MWAIT
100   CALL MTX(IBUF,-1,1)
      CALL MWAIT
      CALL CLOSEC(IDCH)
      CALL RESET
      STOP'PLOT SUCCESSFULLY COMPLETED'
      END
```

**** TAPER ****

PURPOSE	To apply a cosine taper to time data stored.
LANGUAGE	APAL
FORTRAN CALL	CALL TAPER(A,I,B,J,NTAP)
PARAMETERS	A = Source base address I = A address increment B = Destination base address J = B address increment NTAP = Number of elements in taper
FORMULA	B(mJ) = A(mI)*0.5*(cos(m*PI/NTAP)+1) for m=0 to NTAP-1
DESCRIPTION	It use is to avoid the introduction of edges in the data when applying a mute function or prior to a Fourier Transform of the data to the frequency domain to taper both the start and end of the trace to zero to prevent a discontinuity on the wrap round. For a reverse taper set both I and J to -1.
EXAMPLES	See Time Sequence Analysis Program
SRATCH	SP(0 - 4,17), DPX(-1 - 1),DPY(-3 - 0),FA,FM, MD,TM
EXTERNALS	DIV,COS,SPFLT

```
 $TITLE TAPER
 $ENTRY TAPER,5
 $EXT DIV,COS,SPFLT
"
"  WRITTEN: R.W.HOBBS JANUARY 1983
"
"  TO MULTIPLY DATA WITH A COSINE TAPER NTAP SAMPLES LONG
"  STARTING AT 0.0 AND GOING TO 1.0
"
 A $EQU 0 "BASE ADDRESS OF VECTOR TO BE TAPERED
 I $EQU 1 "ADDRESS INCREMENT
 B $EQU 2 "DESTINATION ADDRESS OF TAPERED VECTOR
 J $EQU 3 "ADDRESS INCREMENT
 NTAP $EQU 4 "NUMBER OF ELEMENTS FOR TAPER
 FLT $EQU 17 "LOCATION FOR S-PAD FLOAT COMMAND
"
"
TAPER: SUB I,A;FADD ZERO,ZERO "INITIALIZE COUNTERS ETC.
 SUB J,B;FADD
 LDDPA;DB=ZERO
 MOV NTAP,FLT  "FLOAT INTEGER ELEMENT COUNT
 JSR SPFLT
 LDTMA;DB=|PI  "GET VALUE PI FROM TM
 DPX(0)<DB;DB=DPX(1) "MOVE ELEMENT COUNT
 DPY(0)<TM  "SAVE PI
 DPY(-1)<TM  "SAVE PI
 JSR DIV  "CALCULATE PI/NTAP
 DPX(-1)<DB;DB=DPX(0) "AND SAVE
 LDTMA;DB=|ONE  "GET 1.0 FROM TM
 FSUB ZERO,DPY(-1) "CALCULATE -PI
 FADD;DPY(-2)<TM  "SAVE 1.0
 LDTMA;DB=|HALF  "GET 0.5 FROM TM
 DPY(-1)<FA  "SAVE -PI
 DPY(-3)<TM  "SAVE 0.5
 ADD I,A;SETMA;DPX(0)<DB;DB=DPY(-1) "GET FIRST DATA VALUE,MOVE -PI TO DPX(0)
"
LOOP: JSR COS  "CALCULATE COS(DPX(0))
 FADD DPX(0),DPY(-2) "COS+1
 FADD DPY(-1),DPX(-1) "INCREMENT ELEMENT SUM
 FMUL DPY(-3),FA  "CALCULATE 1/2(COS+1)
 FMUL;FADD
 FMUL;DPY(-1)<FA  "STORE NEW ELEMENT SUM
 DPX(1)<FM  "STORE 1/2(COS+1)
 FMUL DPX(1),MD;ADD I,A;SETMA "SCALE DATA POINT,GET NEXT DATA POINT
 FMUL;DPX(0)<DB;DB=DPY(-1) "MOVE ELEMENT SUM TO DPX(0)
 FMUL
 ADD J,B;SETMA;MI<FM  "STORE SCALED DATA BACK IN MD
 DEC NTAP  "DECREMENT TAPER COUNT
 BGT LOOP  "BRANCH FOR MORE
 RETURN
 $END
```

**** CTAP ****

PURPOSE

To apply a cosine taper to complex numbers.

LANGUAGE

APAL

FORTRAN CALL

```
CALL CTAP(A,B,HP,NTAP)
```

PARAMETERS

```
A    = Source base address
B    = Destination base address
HP   = 0  Taper applied from 1.0 to 0.0
          (Low pass)
       1  Taper applied from 0.0 to 1.0
          (High pass)
NTAP = Number of complex pairs to taper
```

FORMULA

```
B(2*m) = A(2*m)*0.5*(cos(m*PI/NTAP)+1)

B(2*m+1)= A(2*m+1)*0.5*(cos(m*PI/NTAP)+1)

          where X=m          for HP = 1
                X=NTAP-1-m   for HP = 0
            for m=0 to NTAP
```

DESCRIPTION

This is the frequency domain equivalent of TAPER. Both the real and imaginary terms are multiplied by the same factor to so to apply the taper to the amplitude spectrum not the phase spectrum (this becomes undefined when the amplitude is reduced to zero).

EXAMPLE

It is used in the band-pass and band-reject sub-routines. Again it applies a cosine taper to the data except this deals exclusively with complex data.

SRATCH

SP(0 - 4,17),DPX(-1 - 1),DPY(-3 - 0),FA,FM, MD,TM

EXTERNALS

DIV,COS,SPFLT

```
"
 $TITLE CTAP
 $ENTRY CTAP,4
 $EXT DIV,COS,SPFLT
"
"  WRITTEN: R.W.HOBBS JANUARY 1983
"
"  TO APPLY A COSINE TAPER TO A COMPLEX VECTOR WITH EFFECT
"  TO HIGH(HP=1)/LOW(HP=0) PASS IN THE FREQUENCY DOMAIN
"
 A $EQU 0 "BASE ADDRESS OF COMPLEX VECTOR TO BE TAPERED
 B $EQU 1 "DESTINATION ADDRESS OF TAPERED COMPLEX VECTOR
 HP $EQU 2 "HIGH PASS FILTER FLAG
 NTAP $EQU 3 "NUMBER OF COMPLEX ELEMENTS FOR TAPER
 COMP $EQU 4 "COMPLEX FLAG USED IN CODE
 FLT $EQU 17 "LOCATION FOR S-PAD FLOAT COMMAND
"
"
CTAP: DEC A;FADD ZERO,ZERO
 DEC B;FADD
 LDSPI COMP;DB=-1
 LDDPA;DB=ZERO
 MOV NTAP,FLT
 JSR SPFLT
 LDTMA;DB=|PI
 DPX(0)<DB;DB=DPX(1)
 DPY(0)<TM
 DPY(-1)<TM
 JSR DIV
 DPX(-1)<DB;DB=DPX(0)
 LDTMA;DB=|ONE
 NOP
 DPY(-2)<TM
 LDTMA;DB=|HALF
 NOP
 MOV HP,HP;DPY(-3)<TM
 BEQ LOW
HIGH:   FSUB ZERO,DPY(-1)
 FADD
 DPY(-1)<FA
 DPX(0)<FA
 JMP .+2
LOW: DPY(-1)<DB;DB=DPX(-1)
 INC A;SETMA
"
REAL: JSR COS
 FADD DPX(0),DPY(-2)
 FADD DPY(-1),DPX(-1)
 FMUL DPY(-3),FA
 FMUL;FADD
 FMUL;DPY(-1)<FA
 DPX(1)<FM
CMPX: FMUL DPX(1),MD;INC A;SETMA
 FMUL;DPX(0)<DB;DB=DPY(-1)
 FMUL
 INC B;SETMA;MI<FM
 COM COMP
 BGE CMPX
 DEC NTAP
 BGT REAL
 RETURN
 $END
```

**** TVBP ****

PURPOSE — Time varying band-pass function.

LANGUAGE — APAL

FORTRAN CALL — CALL TVBP(A,B,C,D,S,L,N,FLAG)

PARAMETERS

A = Source address of input trace
B = Destination address for output trace
C = Base address of filter coefficients
D = Base address of filter increments
S = Start address of time varying filter
L = Stop address of time varying filter
N = Total number of input samples
FLAG = Filter flag

OPERATION

FLAG = 1 source and destination addresses increment.
Up to address S apply filter described by the filter coefficients at address C by recursion. Between addresses S and L increment filter coefficients at address C by steps at address D each time filter is applied. After address L use final filter coefficients until end of trace.

FLAG =-1 source and destination addresses decrement.
Filter works in in reverse.

DESCRIPTION — Ref: Nikolic (1969).

Using a simple analogue Butterworth band-pass filter designed in the s-plane and tranformed to the z-plane by a bilinear tranformation, to give a single pole, two zero's filter which could be applied by recursion techniques to a time series. Further, by designing the two sets of coefficients to represent the required band-pass at the start and end of the time varying section, and during the filter application incrementing the coefficients from one to the other a true time varying band-pass results. Being an analogue filter it cannot provide total rejection of frequencies outside the pass band region as the digital band-pass routine can, but repeated aplications of the same filter will produce extremely high roll-off rates outside the pass band.

EXAMPLE — Time Sequence Analysis Program

SRATCH — SP(0 - 13),DPX(-4 - 0),DPY(-4 - 0),FA,FM,MD

EXTERNALS — DEQ22

```
"
 $TITLE TVBP
 $ENTRY TVBP,10
 $EXT   DEQ22
"
" WRITTEN BY R.W.HOBBS SEP 1984
" TO EFFECT A TIME VARYING BAND-PASS FUNCTION USING IIR
" BUTTERWORTH FILTER APPLIED BY A RECURSIVE AP ROUTINE DEQ22
"
" INPUT S-PAD PARAMETERS
"
 IA = 0  "START ADDRESS OF I/P TRACE
 IB = 1  "START ADDRESS OF O/P TRACE
 IC = 2  "BASE ADDRESS OF 5 FILTER COEFF.
 ID = 3  "BASE ADDRESS OF 5 COEFF. INCREMENTS
 IS = 4  "START SAMPLE FOR VARYING COEFF.
 IL = 5  "LAST SAMPLE FOR VARYING COEFF.
 IN = 6  "TOTAL NUMBER OF I/P SAMPLES
 FLG = 7  "DIRECTION OF FILTER 1 OR -1
"
" INPUT S-PAD DECLARATONS FOR DEQ22
"
 A = 0  "SOURCE BASE ADDRESS
 I = 1  "I/P ADDRESS INCREMENTS
 B = 2  "BASE ADDRESS OF 5 FILTER COEFF. = IC
 C = 3  "DESTINATION BASE ADDRESS
 K = 4  "O/P ADDRESS INCREMENT
 N = 5  "SAMPLE COUNT
"
" TEMP S-PAD STORAGE
"
 JA = 10  "CURRENT I/P ADDRESS
 JB = 11  "CURRENT O/P ADDRESS
 JD = 12  " = ID
 JS = 13  " = IS
 JL = 14  " = IL
 JN = 15  "LOOP COUNTER
"
TVBP: MOV IA,JA; FADD ZERO,ZERO "SAVE S-PAD PARAMETERS
 MOV IB,JB; FADD
 MOV ID,JD
 MOV IS,JS
 MOV FLG,I    "START SET UP DEQ22 S-PAD
 MOV FLG,K;BGT POS  "IF FLG > 0 GOTO POS
 COM JS     "FIND LOOP COUNT OF START OF TVBP
 ADD IN,JS
 MOV IN,JL    "FIND LOOP COUNT OF END OF TVBP
 SUB IL,JL
 ADD FLG,JA    "PRE DEC I/P AND O/P ADR. IF FLG = -1
 ADD FLG,JB; BR CONT
"
POS: MOV IL,JL
"
CONT: CLR JN     "CLEAR LOOP COUNTER
"
LOOP: MOV JA,A    "SET UP ADDRESSES FOR DEQ22 CALL
 MOV JB,C
 LDSPI N; DB=1    "SET DEQ22 SAMPLE COUNT TO 1
 JSR DEQ22    "O/P SAMPLE
 ADD FLG,JA    "INC. CURRENT ADDRESSES
 ADD FLG,JB
 INC JN     "INC. LOOP COUNT
 SUB# JN,IN    "TEST FOR ALL DONE
 BGT NEXT
 JMP DONE
NEXT: SUB# JN,JS    "TEST FOR START OF TVBP
 BGE LOOP    "LOOP FOR MORE
 SUB# JL,JN    "TEST FOR END OF TVBP
 BGT LOOP    "LOOP FOR MORE
"
```

```
 MOV IC,IC; SETMA  "GET FILTER COEFF.
 SPMDAV; INCMA; DPX(-4)<MD "SAVE IN DPX(-4) TO DPX(0)
 SPMDAV; INCMA; DPX(-3)<MD
 SPMDAV; INCMA; DPX(-2)<MD
 SPMDAV; INCMA; DPX(-1)<MD
 SPMDAV; DPX(0)<MD;
   MOV JD,JD; SETMA  "GET COEFF. INCREMENTS
 SPMDAV; INCMA; DPY(-4)<MD "SAVE IN DPY(-4) TO DPY(0)
 SPMDAV; INCMA; DPY(-3)<MD
 SPMDAV; INCMA; DPY(-2)<MD
 SPMDA; INCMA; DPY(-1)<MD;
   FADD DPX(-4),DPY(-4)  "ADD INCREMENT TO COEFF.
 SPMDA; DPY(0)<MD;
   FADD DPX(-3),DPY(-3)
 FADD DPX(-2),DPY(-2);
   MOV IC,IC; SETMA; MI<FA "LOAD COEFF. BACK INTO MD
 FADD DPX(-1),DPY(-1);
   INCMA; MI<FA
 FADD DPX(0),DPY(0);
   INCMA; MI<FA
 FADD;
   INCMA; MI<FA
 INCMA; MI<FA
"
 JMP LOOP    "LOOP AGAIN
"
DONE: RETURN
"
 $END
```

**** NMO ****

PURPOSE — Apply normal move-out correction

LANGUAGE — APAL

FORTRAN CALL

```
CALL NMO(XADR,MADR,VSA,TSA,ITDEL,ITRSA,ITLIM,
         OTDEL,OTRSA,OTLIM,MUTAD,MUTE)
```

PARAMETERS

XADR = Address of trace offset (m)
MADR = Address of interpolation level M for input trace
VSA = Base address of rms velocity table (m per true sample)
TSA = Base address of corresponding time table (true samples)
ITDEL = Delayin int. samplesto start of input trace. (int. samples)
ITRSA = Base address of input interpolated trace
ITLIM = Number of interpolated samples in input trace
OTDEL = Delayin truesamplesto start of output trace. (true samples)
OTRSA = Base address of output trace
OTLIM = Number of true samples in output trace
MUTAD = Base address of mute normalization trace
MUTE = Number of muted interpolated samples at start of input trace.

FORMULA

$$T_{out}^2 = T_{in}^2 - X^2 / V_{o\ rms}^2$$

DESCRIPTION

The input trace may be resampled at a higher frequency by the interpolation level M. This routine then calculates the move-out corrected trace for the given offset, the chosen output sample being linearly interpolated between the two nearest inptut samples. If the input trace has been muted prior to the NMO correction then by setting MUTAD to an address pointing to the normalization trace, the routine automatically increments this trace where the input sample does not lie in muted region defined by MUTE. If no normalization required set MUTAD to -1.

Original routine written by P.Buhl Lamont-Doherty Geological Observatory, New York, U.S.A..

N.B. All input time scales are in samples
True samples is at output sampling interval
Interpolated samples is output samping interval * M
The final value of TSA must be greater or equal to the last output sample
All TSA values must fall on an interger sample count.
Velocity time table defined at X = 0

EXAMPLES As a move-out routine see Time Sequence Analysis routine.
Used to resample data to any sampling interval see Plot Program

SCRATCH SP(0 - 15),DPX(-4 - 3),DPY(-4 - 3),FA,FM, MD,TM

EXTERNALS DIV,SQRT

```
"
" TO PERFORM NORMAL MOVE-OUT CORRECTION TO A TRACE AT OFFSET X=X
" INTERPOLATED TO A LEVEL M, GIVEN THE VELOCITY/TIME FUNCTION
" AT X=0. THE RESULTING TRACE IS AT TRUE SAMPLE RATE
"
" WRITTEN BY R.W.HOBBS DECEMBER 1983
"
 $TITLE NMO
 $ENTRY NMO,14
 $EXT DIV
 $EXT SQRT
"
" S-PAD PARAMETERS
"
 XADR  = 0 "ADDRESS OF OFFSET X
 MADR  = 1 "ADDRESS OF INTERPOLATION LEVEL FOR INPUT TRACE
 VSA   = 2 "ADDRESS OF VELOCITY TABLE IN M/TRUE SAMPLE
 TSA   = 3 "ADDRESS OF CORRESPONDING TIME TABLE IN TRUE SAMPLES
 ITDEL = 4 "DELAY IN INTERPOLATED SAMPLES TO START OF I/P TRACE
 ITRSA = 5 "START ADDRESS OF I/P TRACE
 ITLIM = 6 "NUMBER OF INTERPOLATED SAMPLES IN I/P TRACE
 OTDEL = 7 "DELAY IN TRUE SAMPLES TO START OF O/P TRACE
 OTRSA = 10 "START ADDRESS OF O/P TRACE
 OTLIM = 11 "NUMBER OF SAMPLES IN O/P TRACE
 MUTAD = 12 "ADDRESS OF NORMALIZATION VECTOR
 MUTE  = 13 "NUMBER OF MUTED INTERPOLATED SAMPLES IN INPUT TRACE
"
" OTHER S-PAD ALLOCATION
"
 NTV   = 12
 IOUT  = 13
 ICLR  = 14
 TONEA = 14
 TRAD  = 15
 NORM  = 0
 MSMP  = 1
"
" D-PAD X ALLOCATION
"
 X2    = -4
 DV    = -3
 W     = -2
 T     = -1
 TNMO  = 2
 VNXT  = 3
"
" D-PAD Y ALLOCATION
"
 M     = -4
 I1    = -3
 INV   = -2
 TNXT  = -1
 T2    = 2
 V     = 3
"
 TMONE = |ONE
"
"
NMO: MOV OTRSA,TRAD    "GET O/P START ADR
 MOV OTLIM,ICLR    "GET O/P SAMPLE COUNT
 DEC TRAD    "PRE-DECREMENT START ADR
CLEAR: INC TRAD; SETMA; MI<DB; DB=ZERO "CLEAR O/P ARRAY
 DEC ICLR
 BGT CLEAR
"
 DEC TSA; LDDPA; DB=ZERO  "PRE-DEC TSA AND SET DPA=0
 DEC OTLIM;    "PRE-DEC O/P SAMPLE COUNT
   DPY(TNXT)<DB; DB=ZERO  "SET FIRST TIME TO ZERO
 LDSPI TONEA; DB=TMONE  "GET ADDRESS OF 1.0
 MOV XADR,XADR; SETMA; CFL1 "GET OFFSET X, AND CLEAR FLAG 1
 SPMDAV; DPX(X2)<MD;   "SAVE OFFSET X
```

```
   MOV VSA,VSA; SETMA  "GET FIRST VELOCITY
 MOV MUTAD,NORM;    "MOVE START ADR. OF NORM. VECTOR
   SPMDAV; DPX(VNXT)<MD;  "SAVE FIRST VELOCITY
     FMUL DPX(X2),DPX(X2) "GET X**2
 FMUL; MOV MADR,MADR; SETMA; "GET INTERPOLATION LEVEL
   BGE .+2    "SKIP NEXT LINE IF NO NORMALIZATION
 SFL1; CLR MUTE    "SET FLAG 1 AND CLEAR MUTE
 FMUL; MOV TONEA,TONEA; SETTMA "GET 1.0 FROM TABLE MEMORY
 DPX(X2)<FM; ADD ITRSA,ITLIM "SAVE X**2 AND GET LAST I/P SAMP ADR
 DPY(M)<MD; DEC ITLIM  "SAVE INTERPOLATION LEVEL
 DPY(I1)<TM; DEC VSA  "SAVE 1.0 AND PRE-DEC VSA
 MOV MUTE,MSMP    "MOVE MUTE SAMPLE COUNT
 BFL1 .+2    "SKIP NEXT LINE IF NO NORMALIZATION
 DEC NORM    "PRE-DEC NORMALIZATION VECTOR ADR.
"
NXTV: INC VSA; SETMA; DPY(V)<DPX(VNXT)"GET NEXT VELOCITY
 SPMDAV; DPX(VNXT)<MD  "SAVE NEXT VELOCITY
 INC TSA; SETMA; DPX(T)<DPY(TNXT)"GET NEXT TIME
 SPMDAV; DPY(TNXT)<MD  "SAVE NEXT TIME
 FSUB DPX(VNXT),DPY(V)  "GET VELOCITY DIFFERANCE
 FSUB DPY(TNXT),DPX(T)  "GET TIME DIFFERANCE
 DPY<FA; FADD    "SAVE DEL V
 DPX<FA; FIX FA    "SAVE DEL T AND FIX DEL T
 FADD
 DPY(INV)<FA    "SAVE FIX(DEL T)
 BFNE .+2    "SKIP IF FIX(DEL T).GT.0
 JMP NXTV    "GET NEXT VELOCITY/TIME PAIR
 JSR DIV     "DPX=DEL V/DEL T
 LDSPI NTV; DB=DPY(INV)  "SAVE SAMPLE COUNT
 DEC NTV; DPX(DV)<DPX  "PRE-DEC COUNTER AND SAVE DV
"
LOOP: FMUL DPY(V),DPY(V); FIXT DPX(T) "GET V**2 AND FIXT(T)
 FMUL DPX(T),DPX(T); FADD "GET T**2
 FMUL; DPY<DPX(X2); DPX(TNMO)<FA "MOVE X**2 TO DPY AND SAVE FIXT(T)
 FMUL; DPX<FM;    "SAVE V**2 IN DPX
   LDSPI IOUT; DB=DPX(TNMO) "SAVE O/P SAMPLE NUMBER
 DPY(T2)<FM; SFL0  "SAVE T**2 AND SET FLAG 0
 JSR DIV     "DPX=X**2/V**2
 FADD DPX,DPY(T2)  "GET T**2+X**2/V**2
 FADD ZERO,ZERO; DPX(W)<ZERO; "CLEAR ADDER AND FRACTION W
   SUB OTDEL,IOUT  "GET RELATIVE O/P SAMPLE NUMBER
 DPX<FA; DPY<ZERO; BGE .+2 "SAVE TIN**2, GOTO NEGAD -VE SAMPLE NO.
 JMP NEGAD
 BFLT .-1    "IF TIN**2.LT.0 GOTO NEGAD
 JSR SQRT
 FMUL DPX,DPY(M)    "GET TIN*M
 FMUL; BFL1 .+2    "SKIP NEXT LINE IF NO NORMALIZATION
 INC NORM; SETMA    "GET NORMALIZATION VALUE
 FMUL
 DPX<FM     "GET ABSOLUTE I/P SAMPLE
 FIXT DPX; DPX(TNMO)<DPX  "GET FIXT(TNMO) AND SAVE TNMO
 FADD; DPX<MD    "SAVE NORMALIZATION VALUE
 DPY<FA; FADD ZERO,FA  "SAVE ABSOLUE I/P SAMPLE NO. AND FLOAT
 FADD; LDSPI TRAD; DB=DPY "MOVE ABS I/P SAMP. NO. TO S-PAD
 FSUB DPX(TNMO),FA;  "GET FRACTION TNMO-FIXT(TNMO)
   SUB ITDEL,TRAD  "GET RELATIVE I/P SAMP. NO.
 FADD; BLT NEGAD    "IF REL. SAMP. NO. -VE GOTO NEGAD
 SUB# TRAD,MSMP    "CHECK IF I/P SAMPLE IS MUTED
 ADD ITRSA,TRAD; SETMA;  "GET I/P SAMPLE DATA
   DPX(W)<FA; BGT NEGAD  "SAVE FRACTION, GOTO NEGAD IF SAMP.MUTED
 SPMDAV; DPY<MD; INCMA  "SAVE SAMPLE, GET NEXT SAMPLE
 CFL0     "CLEAR FLAG 0 FOR GOOD O/P SAMPLE
 SUB# ITLIM,TRAD    "CHECK SAMPLE POINT IN RANGE
 FSUBR DPY,MD; BGE DONE  "GET DIFFERANCE,GOTO DONE IF LAST SAMPLE
"
NEGAD: FADD DPX(T),DPY(I1)  "INCREMENT T
 FADD; FMUL DPX(W),FA;  "GET DIFFERANCE*FRACTION
   SUB# OTLIM,IOUT  "CHECK O/P SAMPLE IN RANGE
 FIX FA; FMUL; BGT DONE  "ROUND-UP T,CHECK O/P SAMPLE IN RANGE
 FMUL; FADD DPY(V),DPX(DV) "INCREMENT V
 FADD FM,DPY; DPX(T)<FA  "ADD DATA FRACTION TO DATA AND SAVE T
 FADD DPX,DPY(I1);  "INC NORM VALUE
```

```
   DPY(V)<FA; BFL0 .+4   "SAVE NEW V,IF NO O/P JUMP 4 LINES
 ADD OTRSA,IOUT; SETMA; MI<FA; "OUTPUT NEW DATA SAMPLE TO O/P TRACE
   FADD
 BFL1 .+2    "SKIP NEXT LINE IF NO NORMALIZATION
 MOV NORM,NORM; SETMA; MI<FA "SAVE NEW NORM VALUE
 DEC NTV; FADD ZERO,DPX(T) "DEC COUNTER,FLOAT NEW T
 FADD
 DPX(T)<FA; BLT .+2   "SAVE NEW TSKIP IF -VE
 JMP LOOP    "GET NEXT OUTPUT SAMPLE
 JMP NXTV    "GET NEXT TIME/VELOCITY PAIR
"
DONE: RETURN
"
 $END
```

**** OMN ****

PURPOSE — Remove normal move-out correction

LANGUAGE — APAL

FORTRAN CALL — CALL OMN(XADR,MADR,VSA,TSA,ITDEL,ITRSA,ITLIM, OTDEL,OTRSA,OTLIM)

PARAMETERS

XADR = Address of trace offset (m)
MADR = Address of interpolation level M for input trace
VSA = Base address of rms velocity table (m per true sample)
TSA = Base address of corresponding time table (true samples)
ITDEL = Delayin int. samplesto start of input trace. (int. samples)
ITRSA = Base address of input interpolated trace
ITLIM = Number of interpolated samples in input trace
OTDEL = Delayin truesamplesto start of output trace. (true samples)
OTRSA = Base address of output trace
OTLIM = Number of true samples in output trace

FORMULA — $T_{out}^2 = T_{in}^2 + X^2 / V_{x\ rms}^2$

DESCRIPTION

The input trace may be resampled at a higher frequency by the interpolation level M. Given the velocity function at X = 0 this routine first calculates the velocity function at the required offset X = X. Then calculates the move-out uncorrected trace at the given offset, the chosen output sample being linearly interpolated between the two nearest inptut samples.

N.B. All input time scales are in samples
True samples is at output sampling interval
Interpolated samples is output sampling interval * M
The final value of TSA must be greater or equal to the last output sample
The TSA values mustfallonan interger sample count.
Velocity time table is defined at X = 0

EXAMPLES — See Time Sequence Analysis routine.

SCRATCH — SP(0 - 15),DPX(-4 - 3),DPY(-4 - 3),FA,FM, MD,TM

EXTERNALS — DIV,SQRT

```
"
" TO PERFORM REVERSE MOVE-OUT CORRECTION TO AN OFFSET X=X FOR A TRACE
" INTERPOLATED TO A LEVEL M, GIVEN THE VELOCITY/TIME FUNCTION
" AT X=0. THE RESULTING TRACE IS AT TRUE SAMPLE RATE
"
" WRITTEN BY R.W.HOBBS DECEMBER 1983
"
 $TITLE OMN
 $ENTRY OMN,12
 $EXT DIV
 $EXT SQRT
"
" S-PAD PARAMETERS
"
 XADR  = 0 "ADDRESS OF OFFSET X
 MADR  = 1 "ADDRESS OF INTERPOLATION LEVEL FOR INPUT TRACE
 VSA   = 2 "ADDRESS OF VELOCITY TABLE IN M/TRUE SAMPLE
 TSA   = 3 "ADDRESS OF CORRESPONDING TIME TABLE IN TRUE SAMPLES
 ITDEL = 4 "DELAY IN INTERPOLATED SAMPLES TO START OF I/P TRACE
 ITRSA = 5 "START ADDRESS OF I/P TRACE
 ITLIM = 6 "NUMBER OF INTERPOLATED SAMPLES IN I/P TRACE
 OTDEL = 7 "DELAY IN TRUE SAMPLES TO START OF O/P TRACE
 OTRSA = 10 "START ADDRESS OF O/P TRACE
 OTLIM = 11 "NUMBER OF SAMPLES IN O/P TRACE
"
" OTHER S-PAD ALLOCATION
"
 NTV   = 12
 IOUT  = 13
 ICLR  = 14
 TONEA = 14
 TRAD  = 15
"
" D-PAD X ALLOCATION
"
 X2    = -4
 DV    = -3
 W     = -2
 T     = -1
 TOMN  = 2
 VNXT  = 3
"
" D-PAD Y ALLOCATION
"
 M     = -4
 I1    = -3
 INV   = -2
 TNXT  = -1
 T2    = 2
 V     = 3
"
 TMONE = |ONE
"
"
OMN: MOV OTRSA,TRAD   "GET START ADR OF O/P TRACE
 MOV OTLIM,ICLR   "GET NUMBER OF SAMPLES IN O/P TRACE
 DEC TRAD   "PRE-DECREMENT START ADR
CLEAR: INC TRAD; SETMA; MI<DB; DB=ZERO "CLEAR O/P ARRAY
 DEC ICLR
 BGT CLEAR
"
 DEC TSA; LDDPA; DB=ZERO  "PRE-DEC TSA AND SET DPA=0
 DEC OTLIM;   "PRE-DEC O/P SAMPLE COUNT
   DPY(TNXT)<DB; DB=ZERO  "SET FIRST TIME TO ZERO
 LDSPI TONEA; DB=TMONE  "GET ADDRESS OF 1.0
 MOV XADR,XADR; SETMA  "GET OFFSET X
 SPMDAV; DPY<MD;   "SAVE OFFSET X
   MOV VSA,VSA; SETMA  "GET FIRST VELOCITY
 SPMDAV; DPX(VNXT)<MD;  "SAVE FIRST VELOCITY
   FMUL DPY,DPY   "GET X**2
 FMUL; MOV MADR,MADR; SETMA; "GET INTERPOLATION LEVEL
```

```
   DPX<DPX(VNXT)   "GET FIRST VELOCITY IN DPX
 FMUL; ADD ITRSA,ITLIM  "GET LAST I/P SAMP ADR
 DPX(X2)<FM; DEC ITLIM  "SAVE X**2 AND PRE DEC ITLIM
 DPY(M)<MD; DEC VSA  "SAVE INTERPOLATION LEVEL
 JSR DIV     "GET DPX=X/V(1)
 FIXT DPX    "GET INTEGER PART
 FADD
 FADD ZERO,FA    "FLOAT(IFIX(T))
 FADD
 DPY(TNXT)<FA    "SAVE NEW TNXT
"
NXTV: INC VSA; SETMA; DPY(V)<DPX(VNXT)"GET NEXT VELOCITY
 SPMDAV; DPX(VNXT)<MD  "SAVE NEXT VELOCITY
 INC TSA; SETMA; DPX(T)<DPY(TNXT);"GET NEXT TIME
   FMUL DPX(VNXT),DPX(VNXT) "GET V**2
 FMUL; SPMDAV; DPY(TNXT)<MD "SAVE NEXT TIME
 FMUL DPY(TNXT),DPY(TNXT) "GET T**2
 FMUL; DPX<FM    "DPX=V**2
 FMUL; DPY<DPX(X2)   "DPY=X**2
 DPY(T2)<FM    "SAVE T**2
 JSR DIV     "DPX=X**2/V**2
 FADD DPY(T2),DPX   "GET T**+X**2/V**2
 FADD
 DPX<FA     "SAVE IN DPX
 JSR SQRT    "DPX=TNEW
 FIXT DPX;    "GET INTEGER PART OF TNEW
   MOV TONEA,TONEA; SETTMA "GET 1.0 FROM TABLE MEMORY
 FADD
 FADD ZERO,FA    "FLOAT(FIXT(TNEW))
 FADD; DPY(I1)<TM   "SAVE 1.0
 DPY(TNXT)<FA    "SAVE FLOAT(FIXT(TNEW))
 FSUB DPX(VNXT),DPY(V)   "GET VELOCITY DIFFERANCE
 FSUB DPY(TNXT),DPX(T)   "GET TIME DIFFERANCE
 DPY<FA; FADD ZERO,DPY(I1) "SAVE DEL V
 DPX<FA; FIX FA    "SAVE DEL T AND FIX DEL T
 BFGE .+4    "SKIP IF SUM +VE
 FABS DPX    "GET ABSOLUTE VALUE
 FSUB ZERO,DPY(I1)   "GET -1.0
 DPX<FA; FIX FA    "SAVE DEL T AND FIX DEL T
 FADD; DPY(I1)<FA   "SAVE TIME INCREMENT
 DPY(INV)<FA    "SAVE FIX(DEL T)
 BFNE .+2    "SKIP IF FIX(DEL T).GT.0
 JMP NXTV    "GET NEXT VELOCITY/TIME PAIR
 JSR DIV     "DPX=DEL V/DEL T
 LDSPI NTV; DB=DPY(INV)   "SAVE SAMPLE COUNT
 DEC NTV; DPX(DV)<DPX   "PRE-DEC COUNTER AND SAVE DV
"
LOOP: FMUL DPY(V),DPY(V); FIXT DPX(T) "GET V**2 AND FIXT(T)
 FMUL DPX(T),DPX(T); FADD "GET T**2
 FMUL; DPY<DPX(X2); DPX(TOMN)<FA "MOVE X**2 TO DPY AND SAVE FIXT(T)
 FMUL; DPX<FM;    "SAVE V**2 IN DPX
   LDSPI IOUT; DB=DPX(TOMN) "SAVE O/P SAMPLE NUMBER
 DPY(T2)<FM; SFL0   "SAVE T**2 AND SET FLAG 0
 JSR DIV     "DPX=X**2/V**2
 FSUB DPY(T2),DPX   "GET T**2-X**2/V**2
 FADD ZERO,ZERO; DPX(W)<ZERO; "CLEAR ADDER AND FRACTION W
   SUB OTDEL,IOUT   "GET RELATIVE O/P SAMPLE NUMBER
 DPX<FA; DPY<ZERO; BGE .+2 "SAVE TIN**2, GOTO NEGAD -VE SAMPLE NO.
 JMP NEGAD
 BFLT .-1    "IF TIN**2.LT.0 GOTO NEGAD
 JSR SQRT
 FMUL DPX,DPY(M)    "GET TIN*M
 FMUL
 FMUL
 DPX<FM     "GET ABSOLUTE I/P SAMPLE
 FIXT DPX; DPX(TOMN)<DPX   "GET FIXT(TOMN) AND SAVE TOMN
 FADD
 DPY<FA; FADD ZERO,FA   "SAVE ABSOLUE I/P SAMPLE NO. AND FLOAT
 FADD; LDSPI TRAD; DB=DPY "MOVE ABS I/P SAMP. NO. TO S-PAD
 FSUB DPX(TOMN),FA;   "GET FRACTION TOMN-FIXT(TOMN)
   SUB ITDEL,TRAD   "GET RELATIVE I/P SAMP. NO.
 FADD; BLT NEGAD    "IF REL. SAMP. NO. -VE GOTO NEGAD
```

```
 ADD ITRSA,TRAD; SETMA;  "GET I/P SAMPLE DATA
   DPX(W)<FA    "GET NEXT DATA POINT AND SAVE FRACTION
 SPWDAV; DPY<MD; INCMA  "SAVE SAMPLE, GET NEXT SAMPLE
 CFL0     "CLEAR FLAG FOR GOOD O/P RANGE
 SUB# ITLIM,TRAD    "CHECK SAMPLE POINT IN RANGE
 FSUBR DPY,MD; BGE DONE  "GET DIFF ,GOTO DONE IF LAST SAMPLE
"
NEGAD: FADD DPX(T),DPY(11)  "INCREMENT T
 FADD; FMUL DPX(W),FA;  "GET DIFFERANCE*FRACTION
   SUB# OTLIM,IOUT  "CHECK O/P SAMPLE IN RANGE
 FIX FA; FMUL; BGT DONE  "ROUND-UP T,CHECK O/P SAMPLE IN RANGE
 FMUL; FADD DPY(V),DPX(DV) "INCREMENT V
 FADD FM,DPY; DPX(T)<FA  "ADD DATA FRACTION TO DATA AND SAVE T
 FADD ZERO,DPX(T); DPY(V)<FA; "FLOAT ROUNDED T,SAVE NEW V
   BFL0 .+2    "SKIP IF FLAG 0 SET
 ADD OTRSA,IOUT; SETMA; MI<FA "OUTPUT NEW DATA SAMPLE TO O/P TRACE
 FADD; DEC NTV    "DEC COUNTER
 DPX(T)<FA; BLT .+2  "SAVE NEW T,SKIP IF -VE
 JMP LOOP    "GET NEXT OUTPUT SAMPLE
 JMP NXTV    "GET NEXT TIME/VELOCITY PAIR
"
DONE: RETURN
"
 $END
```

**** CSQ2D ****

PURPOSE — Two dimensional convolution (correlation) routine for square operators of size not greater than 29 by 29.

LANGUAGE — APAL

FORTRAN CALL — CALL CSQ2D(A,I,B,J,C,K,N,M)

PARAMETERS

A = Base address of operand
I = A address increment
B = Base address of operator for correlation
    Top address of operator for convolution
J = B address increment 1 for correlation
                        -1 for convolution
C = Destination address
K = C address increment
N = Number of samples in operand
M = Dimension of operator < 29
    Sample count for C = N+M-1
    Number of traces in A > M

FORMULA

$$C(n) = \sum_{i=1}^{M} \sum_{j=1}^{M} A(n+(-1)^{S}*i+1-M/2,(-1)^{S}*j+1-M/2).B(i,j)$$

For n = 0 to N-1
For convolution s=1
For correlation s=2

DESCRIPTION

As the array processor is too small to permit effective two-dimensional Fourier transforms on large data sets and the possible routine from FPS was not suitable as it needed large borders of zero samples to avoid wrap round effects wasting processor space, CONV2D. A special routine has been written CSQ2D. The result was a small compact routine that needed no padding of the operand data with zeros but limited the operator size has to be square with a dimension of less than 29 by 29 samples.

EXAMPLE — See Time Sequence Analysis Program

SCRATCH — SP(0 - 13),DPX(0 - 32),DPY(0 - 32),FA,FM, MD

EXTERNALS — none

```
"
 $TITLE CSQ2D
 $ENTRY CSQ2D,10
"
" ROUTINE TO DO 2-D CONVOLUTION (CORRELATION) GIVEN A SQUARE OPERATOR
" M*M SAMPLES ON A INPUT TRACE GATHER OF N SAMPLES LONG BY M TRACES
" WIDE.
"
" WRITTEN BY R.W.HOBBS DECEMBER 1983
"
" S-PAD PARAMETERS
"
 A = 0  "OPERAND BASE ADDRESS
 I = 1  "OPERAND ADR INC
 B = 2  "OPERATOR BASE ADR. BOTTOM FOR CORR. TOP FOR CONV.
 J = 3  "OPERATOR ADR INC +VE FOR CORR. -VE FOR CONV.
 C = 4  "DESTINATION BASE ADR.
 K = 5  "DESTINATION ADR INC
 N = 6  "SAMPLE COUNT OF INPUT TRACE
 M = 7  "DIMENSION OF 2-D OPERATOR
   "SAMPLE COUNT FOR C = N+M-1
   "NUMBER OF TRACES IN A = M
"
" OTHER S-PAD ALLOCATION
"
 IRES  = 10
 IP    = 11
 ILAST = 12
 ICNT  = 13
 TEMP  = 14
 IOP   = 15
"
"
CSQ2D: MOV M,IOP  "SAVE OPERATOR DIMENSION AS INPUT TRACE COUNT
 SUB I,A   "PRE-DECREMENT OPERAND ADR
 SUB J,B   "PRE-DECREMENT OPERATOR ADR
 SUB K,C   "PRE-DECREMENT DESTINATION ADR
 MOV A,ILAST;  "SAVE ADR OF FIRST OPERAND
   SFL0   "SET FLAG 0
"
NEXT: MOV N,ICNT  "GET SAMPLE COUNT
 MOV C,IRES  "GET DESTINATION ADR
 MOV M,TEMP; SETDPA "GET OPERATOR COUNT AND SET DATA PAD ADR
"
LOAD: ADD J,B; SETMA;  "GET OPERATOR
   DPY(3)<DB; DB=ZERO "CLEAR DPY
 DEC TEMP; SETDPA; "DEC COUNTER AND SET DPA
   DPY(2)<DB; DB=ZERO "CLEAR DPY
 SPMDAV;   "WAIT FOR MEMORY
   DPX<DB; DB=MD; "SAVE OPERATOR IN DPX
     BGT LOAD  "LOOP FOR MORE
"
LOOP: ADD I,A; SETMA  "GET DATA SAMPLE
 LDDPA; DB=ZERO  "SET DPA TO ZERO
 MOV M,TEMP  "GET OPERATOR COUNT
 BFL0 .+2  "SKIP NEXT INSTRUCTION IF FIRST TRACE
 ADD I,ILAST;  "INCREMENT ADR OF LAST OPERAND
   SETMA; MI<DB; DB=MD "STORE CURRENT VALUE I.E. SHIFT TR(M) TO TR(M-1)
 DEC TEMP; FADD ZERO,ZERO "PRE-DEC OPERATOR COUNT AND
 DEC TEMP; FMUL DPX,MD;  "MULTIPLY SAMPLE WITH ALL OPERATOR
   FADD ZERO,ZERO;  "VALUES STORED IN DPX
     INCDPA; BEQ END  "AND SUM RESULT WITH TIME
 DEC TEMP; FMUL DPX,MD;  "SHIFTED PREVIOUS RESULTS STORED
   FADD ZERO,ZERO;  "IN DPY
     INCDPA; BEQ END
 DEC TEMP; FMUL DPX,MD; FADD ZERO,ZERO; INCDPA; BEQ END
 DEC TEMP; FMUL DPX,MD; FADD FM,DPY; INCDPA; BEQ END
 DEC TEMP; FMUL DPX,MD; FADD FM,DPY; INCDPA; BEQ END
AGAIN: DEC TEMP; FMUL DPX,MD; FADD FM,DPY; DPY(-3)<FA; INCDPA; BGT AGAIN
END: FMUL; FADD FM,DPY; DPY(-3)<FA; INCDPA
 FMUL; FADD FM,DPY; DPY(-3)<FA; INCDPA
```

```
 FADD FM,DPY; DPY(-3)<FA; INCDPA
 FADD; DPY(-3)<FA; INCDPA
 DPY(-3)<FA;
   ADD K,IRES; SETMA "GET LAST PARTIAL SUM
 LDDPA; DB=2  "SET DPA = 2
 BFL0 .+4  "SKIP IF FIRST TRACE
 FADD DPY,MD  "FORM NEW PARTIAL SUM
 FADD
 DPY<FA   "STORE IN DPY, THEN TRANSFER TO MEMORY
 MOV IRES,IRES; SETMA; MI<DB; DB=DPY
 DEC ICNT  "DECREMENT SAMPLE COUNTER
 BEQ DONE  "GOTO DONE IF LAST SAMPLE IN TRACE
 JMP LOOP  "LOOP FOR NEXT SAMPLE
"
DONE: MOV M,TEMP  "GET OPERATOR COUNT
"
SAVE: ADD K,IRES; SETMA "FLUSH OUT LAST M-1 PARTIAL SUMS
 DEC TEMP; INCDPA
 BEQ CONT  "GOTO CONT WHEN ALL DONE
 BFL0 .+4  "SKIP IF FIRST TRACE
 FADD DPY,MD  "FORM NEW PARTAIL SUMS
 FADD
 DPY<FA   "STORE IN DPY, THEN TRANSFER TO MEMORY
 MOV IRES,IRES; SETMA; MI<DB; DB=DPY;
   BR SAVE  "SAVE ALL PARTIAL SUMS
"
CONT: DEC IOP; CFL0  "DEC TRACE COUNT AND CLEAR FIRST TRACE FLAG
 BEQ EXIT  "WHEN ALL OPERATORS USED GOTO EXIT
 JMP NEXT  "GO FOR NEXT TRACE
"
EXIT: RETURN   "END OF SUBROUTINE
"
 $END
```

**** LINE ****

PURPOSE — To produce logical vector where the values of an input trace cross a defined window or where all values are above the lower window theshold.

LANGUAGE — VECTOR FUNCTION CHAINER (APAL)

FORTRAN CALL — CALL LINE(ITR,ILEVL,ILEVU,I1,I2,I5,IPOL,N, ITYPE)

PARAMETERS

ITR = Base address of input trace
ILEVL = Address of lower window threshold
ILEVU = Address of upper window threshold
I1 = Logical vector workspace 1 length N
I2 = Logical vector workspace 2 length N
I5 = Destination logical vector
IPOL = Plot request, see ITYPE
N = Number of samples in trace
ITYPE = Type of input data

If ITYPE > 0 and IPOL > 0 identify positive values above ILEVL
and IPOL < = 0 plot lines crossing window
If ITYPE < 0 and IPOL < 0 identify negative values below ILEVU
and IPOL > = 0 plot lines crossing window

EXTERNALS — LVGT,LVGE,LVNOT,VOR,WLINE

**** WLINE ****

PURPOSE — To create a logical vector of where a line crosses a given raster window.

LANGUAGE — APAL

FORTRAN CALL — CALL WLINE(A,B,C,N)

PARAMETERS

A = Base address of lower logical vector
B = Base address of upper logical vector
C = Destination logical vector
D = Number of samples

DESCRIPTION — Scans both input logical vectors for where a line crosses into and out of the window. In this region a 1.0 is output to the destination logical array. All other values are set to 0.0.

EXAMPLE — See LINE routine called from Plot program

SCRATCH — SP(0 - 5),DPX(0,1),DPY(-1 - 1)
EXTERNALS — none

```
 DEFINE LINE(ITR,ILEV,ILEV1,I1,I2,I5,IPOL,N,ITYPE)
"
 IF ITYPE < 0 GOTO LT
 CALL LVGT(ITR,1,ILEV,0,I1,1,N)
 IF IPOL > 0 GOTO EXIT
 CALL LVGT(ITR,1,ILEV1,0,I2,1,N)
 GOTO WIGGLE
LT: CALL LVGE(ITR,1,ILEV1,0,I1,1,N)
 CALL LVNOT(I1,1,I1,1,N)
 IF IPOL < 0 GOTO EXIT
 CALL LVGE(ITR,1,ILEV,0,I2,1,N)
 CALL LVNOT(I2,1,I2,1,N)
WIGGLE: CALL WLINE(I1,I2,I1,N)
EXIT: CALL VOR(I1,1,I5,1,I5,1,N)
"
 END



 $TITLE WLINE
 $ENTRY WLINE,4
"
"  ROUTINE TO CREATE LOGICAL VECTOR WHERE A LINE CROSSES A GIVEN
"  RASTER WINDOW
"
"  WRITTEN R.W.HOBBS 1985
"
"  S-PAD PARAMETERS
"
 A = 0  "SOURCE LOGICAL VECTOR
 B = 1  "SOURCE LOGICAL VECTOR
 C = 2  "DESTINATION LOGICAL VECTOR
 N = 3  "NUMBER OF SAMPLES
 IP = 4  "ADVANCE OF ONE ADDRESS
 TONEA = 5 "TM ADDRESS OF 1.0
"
"  DATA PAD NAMES
"
 I0 = 0  "DPX(10)=0.0
 I1 = 1  "DPX(11)=1.0
 M = 1  "DUMP FOR LAG VALUE
 DUMP = -1 "DUMP FOR ADDER OUTPUT
"
 TMONE = 4001
"
WLINE: LDSPI TONEA; DB=TMONE
 MOV A,A; SETMA; FADD ZERO,ZERO
 MOV TONEA,TONEA; SETTMA
 MOV A,IP; FADD; DPX(I0)<DB; DB=ZERO
 INC IP; SPMDAV; DPY(M)<DB; DB=MD
 MOV C,C; SETMA; MI<DPX(I0)
 DEC N; DPX(I1)<TM
 DEC N
"
LOOP: INC IP; SETMA
 SPMDAV; FMUL DPY(M),MD; INC A; SETMA
 FMUL; SPMDA; FADD ZERO,MD; DPY(M)<MD
 FMUL; INC B; SETMA

 FADD FM,ZERO
 SPMDA; FADD ZERO,MD; DPY(DUMP)<FA
 FADD; DPY(DUMP)<FA; BFEQ ZERO
 DPY(DUMP)<FA; BFEQ ONE
 BFEQ ONE
ZERO: INC C; SETMA; MI<DPX(I0)
 BR NEXT
ONE: INC C; SETMA; MI<DPX(I1)
NEXT: DEC N
 BGT LOOP
 INC C; SETMA; MI<DPX(I0)
"
 RETURN
 $END
```

**** LOR ****

PURPOSE	Bit-wise logical OR on 16 bit unsigned integer words stored in lower 16 bits of AP memory
LANGUAGE	APAL
FORTRAN CALL	CALL LOR(A,B,C,N)
PARAMETERS	A = Base address of input integer words B = Base address of input integer words C= Destination address forOR'edinteger words N = Number of input words.
DESCRIPTION	An input word is taken from each input array and OR'ed together in a bit-wise fashion. The result is stored in the destination array.
EXAMPLES	Plot program
SCRATCH	SP(0 - 5),DPX(0),MD
EXTERNALS	none

```
 $TITLE LOR
 $ENTRY LOR,4
"
"  LOGICAL OR FOR INTEGER WORDS STORED IN LOWER 16 BITS OF
"  AP MAIN MEMORY
"
"  WRITTEN R.W.HOBBS 1985
"
" S-PAD DEFINITIONS
"
 A = 0  "SOURCE INPUT LOGICAL ARRAY
 B = 1  "SOURCE INPUT LOGICAL ARRAY
 C = 2  "DESTINATION LOGICAL ARRAY
 N = 3  "SAMPLE COUNT
 PLOT = 4 "PLOT IMAGE WORD
 BACK = 5 "BACKGROUND IMAGE WORD
"
LOR: DEC A; FADD ZERO,ZERO  "PRE DECREMENT A
 DEC B; FADD   "PRE DECREMENT B
 DEC C    "PRE DECREMENT C
 DEC N    "PRE DECREMENT N
"
LOOP: INC A; SETMA   "GET PLOT IMAGE
 SPMDAV; LDSPI PLOT; DB=MD
 INC B; SETMA   "GET BACKGROUND
 SPMDAV; LDSPI BACK; DB=MD
 OR BACK,PLOT;   "LOGICAL OR BACKGROUND AND PLOT
   DPX<DB; DB=SPFN
 INC C; SETMA; MI<DPX  "SAVE OR'ED IMAGE
 DEC N
 BGE LOOP
"
 RETURN
 $END
```

**** RASTER ****

PURPOSE	To compress logical vectors to 16 unsigned interger words
LANGUAGE	APAL
FORTRAN CALL	CALL RASTER(A,B,N)
PARAMETERS	A = Base address og input logical vector B = Destination address for 16 bit words N = Sample count
DESCRIPTION	The routine reads 16 logical numbers from the input vector. If a 1.0 is read the corresponding bit in the output word is set. The routine automatically padds the last word if N is not an integer factor of 16. Output words are byte swapped ready for transfer to Versatec plotter.
EXAMPLES	See Plot program
SCRATCH	SP(0 - 9),DPX(0),MD
EXTERNALS	none

```
 $TITLE RASTER
 $ENTRY RASTER,3
"
" WRITTEN R.W.HOBBS 1984
"
"  HIGH SPEED RASTER CONVERSION
"
 A = 0  "INPUT LOGICAL REAL ARRAY
 B = 1  "OUTPUT RASTER BIT PATTERN AS INTEGERS
 N = 2  "NUMBER OF SAMPLES
 N8 = 3  "8.
 RAS = 4  "RASTER WORD
 N1  = 5  "1.
 N256 = 6 "256.
 TMP = 7  "TEMPORARY STORAGE
 NP = 10  "PLOT MASK
 SWAP = 11 "BYTE SWAP BUFFER
"
RASTER: DEC A
 DEC B
 LDSPI N8; DB=8.
 LDSPI N1; DB=1
 LDSPI N256; DB=400
AGAIN: MOV N8,TMP
 MOV N1,NP
 CLR SWAP
 CLR RAS; CFL0
LOOP: INC A; SETMA
 SPMDAV
 FADD ZERO,MD
 FADD; DEC N
 BLT END
 MOVL RAS,RAS
 BFEQ .+2
 OR NP,RAS
 DEC TMP
 BGT LOOP
 BFL0 OUT
 MOV RAS,SWAP
 MOV N8,TMP
 MOV N256,NP
 CLR RAS; SFL0
 BR LOOP
OUT: OR SWAP,RAS; DB<SPFN; DPX<DB
 INC B; SETMA; MI<DPX
 JMP AGAIN
END: SUB# TMP,N8
 BEQ EXIT
FILL: MOVL RAS,RAS
 DEC TMP
 BGT FILL
 OR SWAP,RAS; DB=SPFN; DPX<DB
 INC B; SETMA; MI<DPX
EXIT: RETURN
 $END
```

**** APHAN ****

PURPOSE	Provide Hanning window smoothing function
LANGUAGE	VECTOR FUNCTION CHAINER (APAL)
FORTRAN CALL	CALL APHAN(A,B,C,NGATE,NSTEP,N)
PRAMETERS	A = Base address of input vector B = Address of workspace for HANN routine C = Destination address NGATE = Size of Hanning window, power of 2 NSTEP = Step increment for calculation N = Input sample count of A Sample count of B = N/NSTEP
DESCRIPTION	Uses AP routines HANN and SVE to calculate the an averaged output value over an NGATE window at every NSTEP interval of input vector. NGATE should be greater than NSTEP.
EXAMPLE	Velocity analysis routine
EXTERNALS	HANN,SVE

```
"
 DEFINE APHAN(ND,ID,IR,NGATE,NSTEP,N)
"
LOOP: CALL HANN(ND,1,ID,1,NGATE,0)
 CALL SVE(ID,1,IR,NGATE)
 IR=IR+1
 ND=ND+NSTEP
 N=N-1
 IF N > 0 GOTO LOOP
 END
```

**** TRREAD ****

PURPOSE — To transfer data from tape to high memory

LANGUAGE — FORTRAN

FORTRAN CALL

```
CALL TRREAD(SEIS,ICDPNO,IWORD,ITRI,ITRO,
                ISTOP)
```

PARAMETERS

SEIS = Real VIRTUAL array of dimension 4156
ICDPNO = Current CMP number
IWORD = Number of words to read
ITRI = Number of traces read in current CMP
ITRO = Number of traces written in current CMP
ISTOP = Error codes
- 11 = Tape read error
- 12 = Non-existant drive
- 13 = Tape header error

DESCRIPTION — DMA transfer to SEIS array. Routine checks for sample count errors. At end of input tape will prompt user for next drive number and checks availablity.

```
      SUBROUTINE TRREAD(SEIS,ICDPNO,IWORD,ITRI,ITRO,ISTOP)
C
      VIRTUAL SEIS(4156),HDRBUF(800)
      REAL*4 TRHDBF(60),TMNAM(3)
      INTEGER*2 TPDRR,TPDRW,ITMERR(2)
      LOGICAL*1 OUTBUF(240),IANS,LDUM
C
      COMMON/DATBUF/OUTBUF
      COMMON/TAPE/  TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
C
      EQUIVALENCE (OUTBUF(29),ITYPE),(OUTBUF(1),TRHDBF(1))
C
C  ASSIGN CHANNEL NUMBER
C
      JCHN=TPDRW+10
      ICHN=TPDRR+10
 10   IRD=IRMTXM(SEIS(1),ICHN,IWORD,ITMERR(1))
      DO 15 I=1,60
 15   TRHDBF(I)=SEIS(I)
      IF(IRD.EQ.IWORD) GO TO 70
      IF(IRD.NE.-1) GO TO 50
      CALL ISPFN ("372,ICHN)
      TYPE 2000,TPDRR
 2000 FORMAT(' END OF READ TAPE ON DRIVE:',I1)
 20   CALL CLOSEC(ICHN)
      TYPE 2100
 2100 FORMAT('$GIVE NEW READ DRIVE NO.:')
      ACCEPT 1000,TPDRR
 1000 FORMAT(I1)
      IF(TPDRR.LE.2) GO TO 30
      ISTOP=12
      RETURN
 30   ICHN=TPDRR+10
      ILK=LOOKUP(ICHN,TMNAM(TPDRR+1),-1)
      ISTUS=IPEEK("172520)
      IF((ISTUS.AND."100).GT.0) GO TO 40
      TYPE 2300,TPDRR
 2300 FORMAT(' GIVEN TAPE DRIVE NO:',I1,' NOT AVAILABLE')
      GO TO 20
C
C  SKIP NEW READ DRIVE HEADER BLOCKS
C
 40   TYPE 2400,TPDRR,(ICDPNO+1)
      WRITE(2,2400) TPDRR,(ICDPNO+1)
 2400 FORMAT(' START NEW READ DRIVE:',I1,' CDP NUMBER:',I5,//)
      IF(IRMTXM(HDRBUF(1),ICHN,1600,ITMERR(1)).NE.1600) GO TO 45
      CALL ISPFNW("376,ICHN)
      IF(IRMTXM(HDRBUF(1),ICHN,200,ITMERR(1)).NE.200) GO TO 45
      CALL ISPFNW("376,ICHN)
      GO TO 10
 45   ISTOP=13
      RETURN
C
C  TRACE READ ERRORS
C
 50   TYPE 2500,TPDRR
 2500 FORMAT(' ERROR ON READ DRIVE:',I1,/,'$CONTINUE(Y/N)?')
      ACCEPT 1100,IANS
 1100 FORMAT(A1)
      ISTOP=11
      IF(IANS.EQ.'Y') GOTO 55
C
C  REWIND READ TAPE TO END OF LAST COMPLETE FILE
C
      CALL BACKSP(ITRI,ICHN)
C
C  REWIND WRITE TAPE TO END OF LAST COMPLETE FILE
C
      IF(ITRO.GT.0) CALL BACKSP(ITRO,JCHN)
      RETURN
 55   ISTOP=0
      ITYPE=2
```

```
C
C  SKIP T.M.
C
 70   CALL ISPFNW("376,ICHN)
C
C  NORMAL RETURN
C
      RETURN
      END
```

**** TRWRIT ****

PURPOSE — To transfer data from high memory to tape

LANGUAGE — FORTRAN

FORTRAN CALL

```
CALL TRWRIT(SEIS,ICDPNO,IWORD,ITRI,ITRO,
                ISTOP)
```

PARAMETERS

SEIS = Real VIRTUAL array of dimension 4156
ICDPNO = Current CMP number
IWORD = Number of words to write
ITRI = Number of traces read in current CMP
ITRO = Number of traces written in current CMP
ISTOP = Error codes
21 = Tape write error
22 = Non-existant drive

DESCRIPTION — DMA transfer from SEIS array. Routine checks for sample count errors. At end of output tape will prompt user for next drive number and checks availablity.

```
      SUBROUTINE TRWRIT(SEIS,ICDPNO,IWORD,FTRCNO,ITRI,ITRO,ISTOP)
C
      VIRTUAL SEIS(4156)
      REAL*4 TRHDBF(60),TMNAM(3)
      INTEGER*4 JCDPTR,JTRCNO
      INTEGER*2 TPDRW,TPDRR,ITMERR(2)
      LOGICAL*1 OUTBUF(240)
C
      COMMON/DATBUF/OUTBUF
      COMMON/TAPE/  TMNAM,TPDRR,TPDRW,IEOTR,IEOTW,ITMERR
C
      EQUIVALENCE (JTRCNO,OUTBUF(1)),(TRHDBF(1),OUTBUF(1)),
     +(JCDPTR,OUTBUF(25))
C
      ITRY=0
C
C     ASSIGN CHANNEL NUMBERS
C
      ICHN=TPDRW+10
      JCHN=TPDRR+10
C
C       WRITE TRACE SEQUENCE NUMBER IN HEADER BLOCK
C
      CALL JICVT(ITRO,JCDPTR)
      CALL JJCVT(JCDPTR)
      CALL JAFIX(FTRCNO,JTRCNO)
      CALL JJCVT(JTRCNO)
      DO 205 I=1,60
 205  SEIS(I)=TRHDBF(I)
 220  IWT=IWMTXM(SEIS(1),ICHN,IWORD,ITMERR(1))
C
C       CHECK FOR ERRORS
C
      IF(IWT.EQ.IWORD) GOTO 210
      IF(IWT.EQ.-1) GOTO 400
C
C       RE-TRY
C
  240 ITRY=ITRY+1
      CALL BACKSP(1,ICHN)
      WRITE(2,2200) TPDRW,ICDPNO,ITMERR(1)
      TYPE 2200,TPDRW,ICDPNO,ITMERR(1)
 2200 FORMAT(' ERROR ON DRIVE:',I2,' CDP NO:',I5,' STATUS:',I5,//)
      IF(ITRY.LT.3) GO TO 220
C
C       BAD OUTPUT FILE SO REWIND TO LAST FILE AND CLOSE TAPE
C
      CALL BACKSP(ITRO,ICHN)
      FTRCNO=FTRCNO-FLOAT(ITRO)+1.0
      CALL ISPFNW("377,ICHN)
C
C  REWIND READ TAPE TO END OF LAST COMPLETE FILE
C
      IF(ITRI.GT.0) CALL BACKSP((ITRI+1),JCHN)
      ISTOP=21
      RETURN
C
C       COME HERE TO WRITE TAPE MARK
C
 210  CALL ISPFNW("377,ICHN)
      ISTUS=IPEEK("172520)
      IF((ISTUS.AND."40000).LE.0) GO TO 240
      IF((ISTUS.AND."2000).EQ.0) GOTO 200
      CALL ISPFNW("375,ICHN)
      GOTO 400
C
C       NORMAL RETURN
C
 200  FTRCNO=FTRCNO+1.0
      RETURN
C
C       COME HERE FOR EOT
```

```
C
 400  IEOTW=1
      CALL BACKSP(ITRO,ICHN)
      FTRCNO=FTRCNO-FLOAT(ITRO)+1.0
C
C       CLOSE REWIND OUTPUT TAPE
C
      CALL ISPFNW("377,ICHN)
      CALL ISPFN ("372,ICHN)
      CALL CLOSEC(ICHN)
C
C  REWIND READ TAPE TO BEGINNING OF CURRENT FILE
C
      IF(ITRI.GT.0) CALL BACKSP((ITRI+1),JCHN)
      TYPE 2300,TPDRW,(ICDPNO-1)
      WRITE(2,2300) TPDRW,(ICDPNO-1)
 2300 FORMAT(' EOT ON WRITE DRIVE:',I2,' CDP NO:',I5,)
      TYPE 2310
 2310 FORMAT('$GIVE NEW WRITE DRIVE NUMBER:')
      ACCEPT 1000,TPDRW
 1000 FORMAT(I1)
      IF(TPDRW.LE.2) RETURN
      ISTOP=22
      RETURN
      END
```

**** EXMLIB ****

PURPOSE

To provide simple fortran call to give extended memory DMA facilities.

LANGUAGE

PDP-11 MACRO

FORTRAN CALL

```
CALL IRDKXM(VARRAY(I),CHN.NO.,N,BLOCK,ISYN)
CALL IWDKXM(VARRAY(I),CHN.NO.,N,BLOCK,ISYN)
CALL IRDRXM(VARRAY(I),CHN.NO.,N,BLOCK,ISYN)
CALL IWDRXM(VARRAY(I),CHN.NO.,N,BLOCK,ISYN)
CALL APPTXM(VARRAY(I),AP.ADDR.,N,TYPE)
CALL APGTXM(VARRAY(I),AP.ADDR.,N,TYPE)
CALL IRMTXXM(VARRAY(I),CHN.NO.,N,ITMERR(2))
CALL IWMTXXM(VARRAY(I),CHN.NO.,N,ITMERR(2))
CALL IRMTXLM( ARRAY(I),CHN.NO.,N,ITMERR(2))
CALL IWMTXLM( ARRAY(I),CHN.NO.,N,ITMERR(2))
```

PARAMETERS

```
VARRAY(I) = VIRTUAL array name and location
CHN.NO. = RT-11 channel number
N = Sample count
BLOCK = RT-11 file block number at start of
          transfer
ISYN = 0 synchronous transfer, routine
          waits until complete.
     = 1 asynchronous transfer, routine
          returns without waiting.
AP.ADDR. = AP. start address
TYPE = Format type 1 integer
                   2 DEC real
                   3 IBM real
ARRAY(I) = Low memory array
ITMERR(2) = Two element array for the
          return of error codes from tape
          controller.
```

DESCRIPTION

```
IR.... stands for a read routine
..DK.. Pertec DK drive transfers
..DR.. Whinchester drive transfers
..MT.. tape drive transfers
....XM extended memory (VIRTUAL)
....LM low memory (REAL)
APPT.. puts data into array processor
APGT.. gets data from array processor
```

These routines have been developed from work done by Poulter (1982) and Wellby (1983). They allow data transfers to VIRTUAL memory, except for ....LM tape routines that provide low memory support. All the routines modify the extended memory bits in the device handlers, these are then cleared by completion routines.

```
 .TITLE EXMLIB
 .MCALL .READW,.WRITW,.WRITC,.READC,.SPFUN
 .ENABL GBL
 .GLOBL IRDKXM,IWDKXM,APPTXM,APGTXM,IRMTXM,IWMTXM,IWMTLM,IRMTLM
 .GLOBL IRDRXM,IWDRXM
 .GLOBL RUNXMA,APWD


;CALLING CALL IRDKXM(VARRAY(I),CHN.NO.,WORD COUNT,BLOCK NO.,ISYNCH)
;CALLING CALL IWDKXM(VARRAY(I),CHN.NO.,WORD COUNT,BLOCK NO.,ISYNCH)
;CALLING CALL IRDRXM(VARRAY(I),CHN.NO.,WORD COUNT,BLOCK NO.,ISYNCH)
;CALLING CALL IWDRXM(VARRAY(I),CHN.NO.,WORD COUNT,BLOCK NO.,ISYNCH)
;CALLING CALL APPTXM(VARRAY(I),AP.ADDRESS,COUNT,TYPE)
;CALLING CALL APGTXM(VARRAY(I),AP.ADDRESS,COUNT,TYPE)
;CALLING CALL IRMTXM(VARRAY(I),CHN.NO.,WORD COUNT,ITMERR(2))
;CALLING CALL IWMTXM(VARRAY(I),CHN.NO.,WORD COUNT,ITMERR(2))
;CALLING CALL IRMTLM(ARRAY(I),CHN.NO.,WORD COUNT,ITMERR(2))
;CALLING CALL IWMTLM(ARRAY(I),CHN.NO.,WORD COUNT,ITMERR(2))


USP0=177640
RKCS=177404
RMCS1=176700
MTCS=172522
CONTRL=176104
LITES=176114
EOF=-1.
HRDWER=-2.
CHNNA=-3.
ERRBYT=52

IRDKXM: MOV @#USP0,R0  ;GET EXTENSION IN R0
 ASL R0   ;GET HIGHBITS INTO PLACE
 ASL R0
 MOV R0,R3   ;SAVE IN R3
 ASL R0   ;GET LOWBITS INTO PLACE
 ASL R0
 ASL R0
 ASL R0
 BIC #177700,R1  ;CLEAR UNWANTED BITS
 BIC #77,R0   ;IN R1 + R0
 BIS R1,R0   ;SET LOWBITS
 SWAB R3   ;GET HIGHBITS INTO
 BIC #177717,R3  ;CORRECT PLACE
 MOV R0,R2   ;FREE R0 FOR MACRO CALL


1$: TST @#RKCS   ;WAIT FOR DISC CONTROLLER
 BPL 2$
 BR 1$
2$: TSTB @#RKCS
 BPL 1$


 BIC #60,@#RKCS  ;CLEAR EXMEM BITS
 BIS R3,@#RKCS  ;SET EXMEM BITS TO REQUIRED VALUE
 MOV @#4(R5),R1  ;GET ARGUMENTS INTO
 MOV @#6(R5),R3  ;INTO REGISTERS
 MOV @#10(R5),R4
 CMP @#12(R5),#1  ;ASYNCHRONOUS READ?
 BNE 3$   ;BRANCH FOR WAIT READ.
 .READC #AREA,R1,R2,R3,#RDFIN,R4
 BCC RETPC   ;BRANCH IF NO ERRORS. XMEM
     ;BITS WILL BE CLEARED BY
     ;COMPLETION ROUTINE.

 JSR PC,DSKERR  ;DETERMINE ERROR VALUE TO BE
     ;PASSED BACK TO FORTRAN.

 TST R0   ;TEST FOR PARTIAL READ.
```

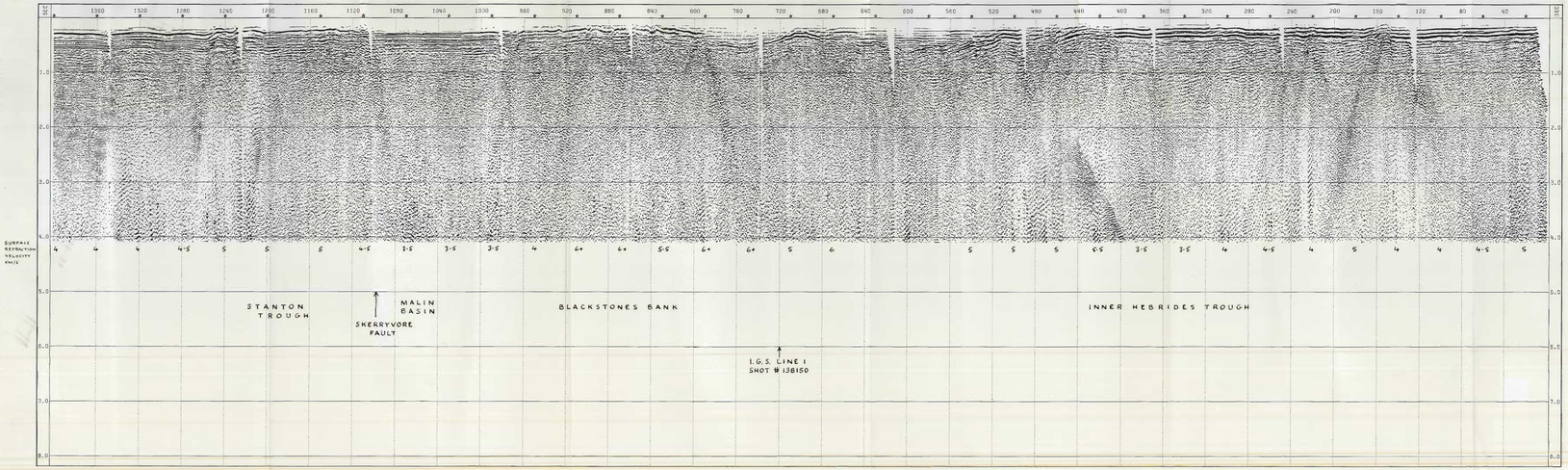
```
 BMI RDFIN    ;BRANCH IF NO READ IS TO OCCUR
      ;BECAUSE COMPLETION ROUTINE WILL
      ;NOT BE CALLED IN THESE
      ;CIRCUMSTANCES & SO XMEM BITS
      ;MUST BE CLEARED NOW.
 BR RETPC    ;COMPLETION ROUTINE WILL CLEAR
      ;XMEM BITS IF PARTIAL READ OCCURS.

3$: .READW #AREA,R1,R2,R3,R4   ;START READ
 BCC RDFIN    ;BRANCH IF NO ERRORS RETURNED
 JSR PC,DSKERR   ;DETERMINE ERROR VALUE TO BE
      ;PASSED BACK TO FORTRAN
RDFIN: BIC #60,@#RKCS   ;CLEAR EXMEM BITS
RETPC: RTS PC

IWDKXM: MOV @#USP0,R0   ;GET EXTENSION IN R0
 ASL R0    ;GET HIGHBITS INTO PLACE
 ASL R0
 MOV R0,R3    ;SAVE IN R3
 ASL R0    ;GET LOWBITS INTO PLACE
 ASL R0
 ASL R0
 ASL R0
 BIC #177700,R1   ;CLEAR UNWANTED BITS
 BIC #77,R0    ;IN R1 + R0
 BIS R1,R0    ;SET LOWBITS
 SWAB R3    ;GET HIGHBITS INTO
 BIC #177717,R3   ;CORRECT PLACE
 MOV R0,R2    ;FREE R0 FOR MACRO CALL


1$: TST @#RKCS    ;WAIT FOR DISC CONTROLLER
 BPL 2$
 BR 1$
2$: TSTB @#RKCS
 BPL 1$


 BIC #60,@#RKCS   ;CLEAR EXMEM BITS
 BIS R3,@#RKCS   ;SET EXMEM BITS TO REQUIRED VALUE
 MOV @#4(R5),R1
 MOV @#6(R5),R3
 MOV @#10(R5),R4
 CMP @#12(R5),#1   ;ASYNCHRONOUS WRITE?
 BNE 3$    ;BRANCH FOR WAIT WRITE.
 .WRITC #AREA,R1,R2,R3,#WRFIN,R4
 BCC PCRET    ;BRANCH IF NO ERRORS. XMEM
      ;BITS WILL BE CLEARED BY
      ;COMPLETION ROUTINE.

 JSR PC,DSKERR   ;DETERMINE ERROR VALUE TO BE
      ;PASSED BACK TO FORTRAN.

 TST R0    ;TEST FOR PARTIAL WRITE.
 BMI WRFIN    ;BRANCH IF NO WRITE IS TO OCCUR
      ;SINCE THE COMPLETION ROUTINE
      ;IS NOT CALLED IN THESE
      ;CIRCUMSTANCES AND SO THE XMEM
      ;BITS MUST BE CLEARED NOW.
 BR PCRET    ;COMPLETION ROUTINE WILL CLEAR
      ;XMEM BITS IF PARTIAL WRITE OCCURS.

3$: .WRITW #AREA,R1,R2,R3,R4
 BCC WRFIN    ;BRANCH IF NO ERRORS RETURNED
 JSR PC,DSKERR   ;DETERMINE ERROR VALUE TO BE
      ;PASSED BACK TO FORTRAN
WRFIN: BIC #60,@#RKCS   ;CLEAR EXMEM BITS
PCRET: RTS PC


DSKERR: CMPB @#ERRBYT,#1
 BMI 9$
```

```
 BEQ 7$
 MOV #CHNNA,R0
 BR 6$
9$: CMP R0,#0  ;CHECK FOR REDUCED WORD COUNT
 BNE 6$  ;BRANCH FOR IMPLIED EOF (PARTIAL TRANSFER)
 MOV #EOF,R0  ;RETURN EOF FLAG IF WORD COUNT IS ZERO
 BR 6$
7$: MOV #HRDWER,R0
6$: RTS PC


IRDRXM: MOV @#USP0,R0  ;GET PAR 0 BLOCK OFFSET
 MOV R0,R3
 ASL R0
 ASL R0
 ASL R0
 ASL R0
 ASL R0
 ASL R0
 BIC #177700,R1
 BIC #77,R0
 BIS R1,R0
 MOV R0,R2
 ASR R3
 ASR R3
 BIC #176377,R3


1$: TSTB @#RMCS1   ;WAIT FOR DISC CONTROLLER
 BEQ 1$


 BIC #1400,@#RMCS1  ;CLEAR XMEM BITS
 BIS R3,@#RMCS1  ;SET XMEM BITS
 MOV @4(R5),R1  ;GET ARGUMENTS
 MOV @6(R5),R3  ;FROM FORTRAN
 MOV @10(R5),R4  ;INTO REGISTERS
 CMP @12(R5),#1  ;ASYNCHRONOUS READ?
 BNE 2$   ;BRANCH FOR WAIT READ
 .READC #AREA,R1,R2,R3,#RDDONE,R4
 BCC RTSPC   ;BRANCH IF NO ERRORS. XMEM
     ;BITS WILL BE CLEARED BY
     ;COMPLETION ROUTINE.

 JSR PC,DSKERR  ;DETERMINE ERROR VALUE TO BE
     ;PASSED BACK TO FORTRAN.

 TST R0   ;TEST FOR PARTIAL READ.
 BMI RDDONE   ;BRANCH IF NO READ IS TO OCCUR
     ;BECAUSE COMPLETION ROUTINE WILL
     ;NOT BE CALLED IN THESE
     ;CIRCUMSTANCES & SO XMEM BITS
     ;MUST BE CLEARED NOW.
 BR RTSPC   ;COMPLETION ROUTINE WILL CLEAR
     ;XMEM BITS IF PARTIAL READ OCCURS.

2$: .READW #AREA,R1,R2,R3,R4 ;DO READ
 BCC RDDONE   ;BRANCH IF NO ERRORS RETURNED
 JSR PC,DSKERR  ;DETERMINE ERROR VALUE TO BE
     ;PASSED BACK TO FORTRAN
RDDONE: BIC #1400,@#RMCS1  ;CLEAR XMEM BITS
RTSPC: RTS PC


IWDRXM: MOV @#USP0,R0  ;GET PAR 0 BLOCK OFFSET
 MOV R0,R3
 ASL R0
 ASL R0
 ASL R0
 ASL R0
 ASL R0
 ASL R0
```

```
 BIC #177700,R1
 BIC #77,R0
 BIS R1,R0
 MOV R0,R2
 ASR R3
 ASR R3
 BIC #176377,R3


1$: TSTB @#RMCS1    ;WAIT FOR DISC CONTROLLER
 BEQ 1$


 BIC #1400,@#RMCS1   ;CLEAR XMEM BITS
 BIS R3,@#RMCS1   ;SET XMEM BITS
 MOV @4(R5),R1   ;GET ARGUMENTS
 MOV @6(R5),R3   ;FROM FORTRAN
 MOV @10(R5),R4   ;INTO REGISTERS
 CMP @12(R5),#1   ;ASYNCHRONOUS WRITE?
 BNE 2$    ;BRANCH FOR WAIT WRITE
 .WRITC #AREA,R1,R2,R3,#WRDONE,R4
 BCC PCRTS    ;BRANCH IF NO ERRORS. XMEM
      ;BITS WILL BE CLEARD BY
      ;COMPLETION ROUTINE.

 JSR PC,DSKERR   ;DTERMINE ERROR VALUE TO BE
      ;PASSED BACK TO FORTRAN.

 TST R0    ;TEST FOR PARTIAL WRITE.
 BMI WRDONE    ;BRANCH IF NO WRITE IS TO OCCUR
      ;SINCE THE COMPLETION ROUTINE
      ;IS NOT CALLED IN THESE
      ;CICUMSTANCES AND SO THE XMEM
      ;BITS MUST BE CLEARED NOW.
 BR PCRTS    ;COMPLETION ROUTINE WILL
      ;CLEAR XMEM BITS IF
      ;PARTIAL WRITE OCCURS.

2$: .WRITW #AREA,R1,R2,R3,R4 ;DO READ
 BCC WRDONE    ;BRANCH IF NO ERRORS RETURNED
 JSR PC,DSKERR   ;DETERMINE ERROR VALUE TO BE
      ;PASSED BACK TO FORTRAN
WRDONE: BIC #1400,@#RMCS1   ;CLEAR XMEM BITS
PCRTS: RTS PC



APPTXM: MOV #193.,R3
 BR COMMON
APGTXM: MOV #225.,R3
COMMON: MOV @#USP0,R0
 ASL R0
 ASL R0    ;GET HIGHBITS INTO PLACE
 ASL R0
 ASL R0
 MOV R0,R2    ;SAVE INTO HIGHBITS
 BIC #37777,R2   ;CLEAR OTHER BITS
 ASL R0    ;SET UP 16 BIT PART
 ASL R0
 BIC #177700,R1   ;CLEAR UNWANTED BITS
 BIC #77,R0
 BIS R1,R0    ;FORM 16 BIT PART

 ADD @10(R5),R3
 ADD @10(R5),R3
 MOV R0,ARG1
 MOV 4(R5),ARG2
 MOV 6(R5),ARG3
 MOV R3,CTRL
 MOV #CTRL,ARG4
 MOV R2,ARG5
 MOV R5,-(SP)
```

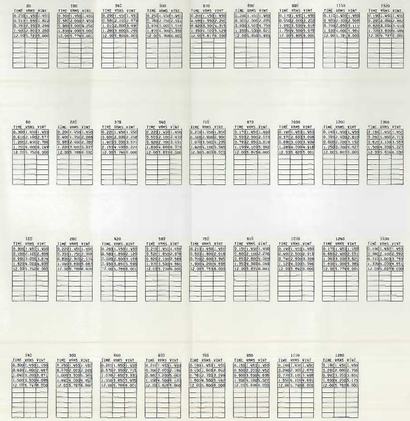
```
 MOV #ARGBLK,R5
 JSR PC,RUNXMA
 MOV (SP)+,R5
 RTS PC


ARGBLK: .WORD 5
ARG1: .WORD 0
ARG2: .WORD 0
ARG3: .WORD 0
ARG4: .WORD 0
ARG5: .WORD 0
CTRL: .WORD 0


IRMTLM: MOV 2(R5),R2
 TSTB @#MTCS
 BEQ IRMTLM
 BIC #60,@#MTCS
 BR CONT1
IRMTXM: MOV @#USP0,R0
 ASL R0
 ASL R0
 MOV R0,R3   ;SAVE XMEM BITS IN R3
 ASL R0
 ASL R0
 ASL R0
 ASL R0
 BIC #177700,R1  ;CLEAR UNWANTED BITS
 BIC #77,R0   ;IN BOTH R0 AND R1
 BIS R1,R0   ;SET UP ADDRESS IN R0
 SWAB R3   ;MOV XMEM BITS INTO PLACE
 BIC #177717,R3  ;CLEAR UNWANTED BITS
 MOV R0,R2   ;FREE R0 FOR MACRO CALL

1$: TSTB @#MTCS   ;WAIT FOR TAPE CONTROLLER
 BEQ 1$

 BIC #60,@#MTCS  ;CLEAR XMEM BITS
 BIS R3,@#MTCS  ;SET TO REQUIRED VALUE
CONT1: MOV @4(R5),R1  ;GET ARGUMENTS INTO REGISTERS
 MOV @6(R5),R3
; .READW #AREA,R1,R2,R3,#0 ;.READX IGNORES EOT
  .SPFUN #AREA,R1,#370,R2,R3,#SPACE,#0 ;START READ. USE SPFUN
      ;TO DETECT EOT
 BCC 2$   ;BRANCH IF NO ERRORS RETURNED
 JSR PC,TAPERR  ;DETERMINE ERROR VALUE TO BE
     ;PASSED TO FORTRAN
2$: BIC #60,@#MTCS  ;CLEAR EXMEM BITS
 RTS PC


IWMTLM: MOV 2(R5),R2
 TSTB @#MTCS
 BEQ IWMTLM
 BIC #60,@#MTCS
 BR CONT
IWMTXM: MOV @#USP0,R0
 ASL R0
 ASL R0
 MOV R0,R3   ;SAVE XMEM BITS IN R3
 ASL R0
 ASL R0
 ASL R0
 ASL R0
 BIC #177700,R1  ;CLEAR UNWANTED BITS
 BIC #77,R0   ;IN BOTH R0 AND R1
 BIS R1,R0   ;SET UP ADDRESS IN R0
 SWAB R3   ;MOVE XMEM BITS INTO PLACE
 BIC #177717,R3  ;CLEAR UNWANTED BITS
 MOV R0,R2   ;FREE R0 FOR MACRO CALL
```

```
1$: TSTB @#MTCS    ;WAIT FOR TAPE CONTROLLER
 BEQ 1$

 BIC #60,@#MTCS   ;CLEAR XMEM BITS
 BIS R3,@#MTCS   ;SET TO REQUIRED VALUE
CONT: MOV @4(R5),R1   ;GET ARGUMENTS INTO REGISTERS
 MOV @6(R5),R3
;  .WRITW #AREA,R1,R2,R3,#0 ;WRITX DONT GIVE
      ;FULL ERROR REPORTS
 .SPFUN #AREA,R1,#371,R2,R3,#SPACE,#0 ;START WRITE. USE SPFUN TO
       ;GET FULL ERROR REPORT
 BCC 2$    ;BRANCH IF NO ERRORS RETURNED
 JSR PC,TAPERR   ;DETERMINE ERROR VALUE TO BE
      ;PASSED TO FORTRAN
2$: BIC #60,@#MTCS   ;CLEAR EXMEM BITS
 RTS PC

TAPERR: ADD #2,SP
 MOV 10(R5),R3
 CMPB @#ERRBYT,#1
 BMI 9$
 BEQ 7$
 MOV #CHNNA,R0
 BR 6$
9$: MOV #EOF,R0
 BR 8$
7$: MOV #HRDWER,R0
8$: MOV #SPACE,R2
 MOV (R2)+,(R3)+
 MOV (R2),(R3)
6$: RTS PC


 .EVEN
AREA: .BLKW 10
SPACE: .WORD 0,0,0,0

 .END
 .TITLE DKDRXM
```



FARNELLA 1/81 LINE : 6 START GATHER NO.. 1 END GATHER NO.. 3400
POLARITY REVERSED NEGATIVE LOBE SHADED HIGH CLIP LEVEL 0.050 INCHES
TRACE SPACING 0.125 INCHES SECTION PLOTTED AT 1.25 SECS/INCH
PLOTTING GAIN 7.500E+01 LOW CLIP LEVEL 0.005 INCHES

SURFACE REFRACTION VELOCITY KM/S



DURHAM UNIVERSITY
SEISMIC PROCESSING
FARNELLA 1/81 LINE : 6

SYSTEM PARAMETERS

FIELD DATA		
COLLECTED ON	FARNELLA CRUISE NO: 1/81	
SOURCE/RECEIVER OFFSET	245 METRES	
RECEIVER SPACING	100 METRES	
SAMPLING INTERVAL	4 MSEC	
DEMULTIPLEX		
SORT		
FOLD	24	
COMMON MID-POINT SPACING	50 METRES	
RECORDER ERROR REMOVAL		
DBS		
TIME BREAK CORRECTION	64 MSEC	
SOURCE DEPTH	10 METRES	
RECEIVER DEPTH	10 METRES	
POLARITY REVERSAL TRACES	2,5,10 AND 12	
BAND-PASS FILTER	5,10 TO 40,60 HZ	
VELOCITY FILTER	OPERATOR SIZE 11*41	
	FALSE VELOCITY STRUCTURE	
	TIME (MSEC)	RMS VELOCITY (KM/SEC)
	400	1.5
	1000	2.1
	2000	3.0
MUTE	TRACE NUMBER	TIME (MSEC)
	1	500
	7	1100
	24	2000
STACK		
RESAMPLED PRIOR TO NMO	1 MSEC	
NMO AND STACK	24 FOLD	
DBS		
BAND-PASS FILTER	5,10 TO 40,60 HZ	
GEOMETRIC SPREADING CORRECTION		
DECONVOLUTION	PREDICTION ERROR	
	AUTOCORR. DESIGN	2000 MSEC FROM 0 MSEC
	ACTIVE LENGTH	400 MSEC
	PREDICTION LAG	100 MSEC
	PRE-WHITENING	1 %
TRACE NORMALIZATION	UNIT ENERGY	
DISPLAY		
GAIN	75	
CLIP	0.005 TO 0.05 INCHES	
TRACE SPACING	0.125 INCHES	
VERTICAL SCALE	1.25 INCHES/SEC	
PROCESSED BY	R.W.HOBBS	

TERTIARY PLUTONIC CENTRES AND SILLS

TERTIARY LAVAS

CALEDONIAN GRANITE

MESOZOIC

MOINIAN

TORRIDONIAN

LEWISIAN

SEA OF THE HEBRIDES REGION

TRANSVERSE MERCATOR PROJECTION

SCALE 1:250000

SUB-CROP GEOLOGY

8°W 7°W 6°W

57°N 56°30' 56°N

OUTER ISLES THRUST ZONE

BARRA

BARRA TROUGH

OUTER ISLES PLATFORM

SEA of the HEBRIDES TROUGH

tertiary sediments

CANNA RIDGE

RHUM

CAMASUNARY FAULT

MOINE THRUST ZONE

SLEAT

HAWES BANK

COLL BASIN

COLL

MINCH FAULT

TIREE

TIREE PLATFORM

MULL

STANTON TROUGH

INNER HEBRIDES TROUGH

SKERRYVORE FAULT

? MOINE THRUST

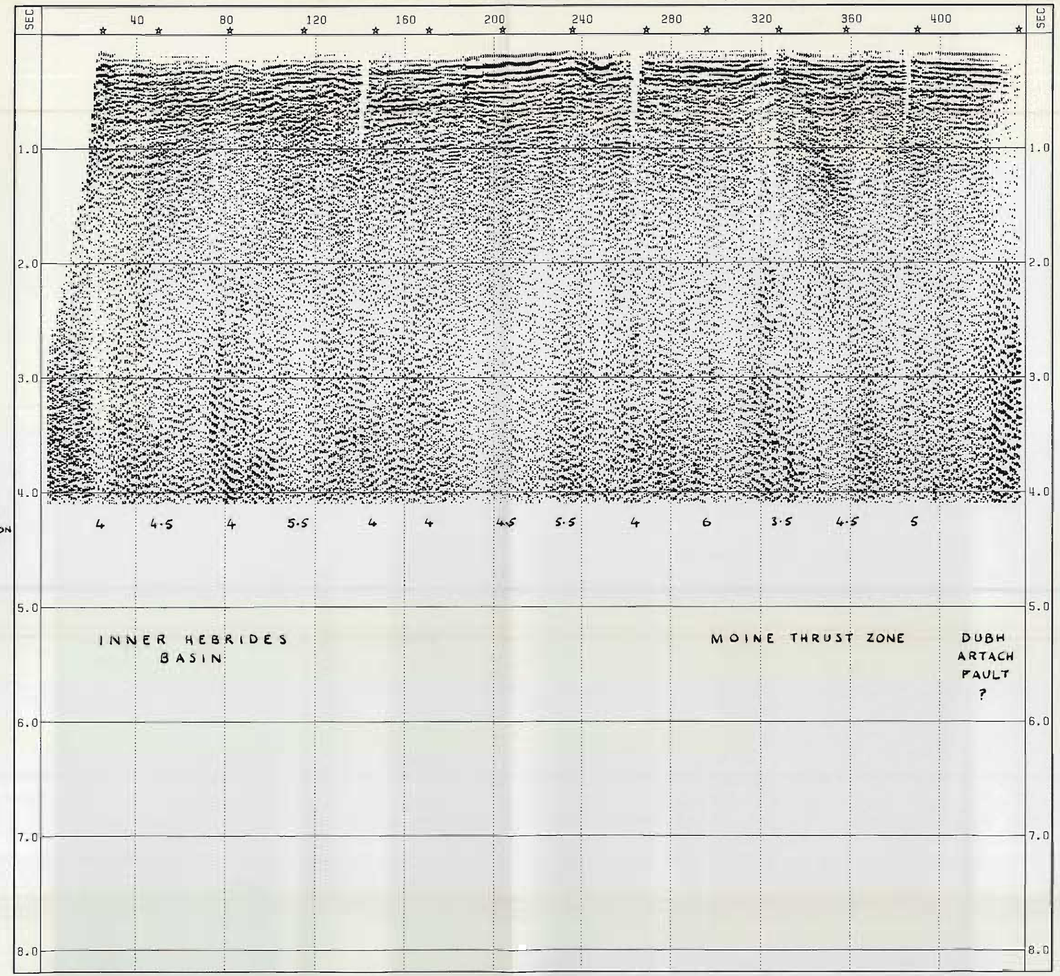
DUBH ARTACH FAULT

BLACKSTONES BANK

STANTON BANKS

MALIN BASIN

JURA



FARNELLA 1/81 LINE : 4 START GATHER NO. 1 END GATHER NO. 435
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.025 INCHES TRACE LENGTH 5.12 INCHES
PLOTTING GAIN 7.500E+01 LOW CLIP LEVEL 0.005 INCHES HIGH CLIP LEVEL 0.050 INCHES

25

TIME	VRMS	VINT
0.100	1.450	1.450
0.370	1.800	1.913
0.530	2.400	3.409
0.750	3.600	5.505
8.000	5.817	6.000

147

TIME	VRMS	VINT
0.080	1.450	1.450
0.220	1.700	1.828
0.700	2.500	2.791
0.850	3.150	5.202
8.000	5.764	5.999

269

TIME	VRMS	VINT
0.100	1.450	1.450
0.200	1.850	1.828
0.480	2.250	2.595
0.950	3.300	4.104
8.000	5.746	6.000

390

TIME	VRMS	VINT
0.100	1.450	1.450
0.220	1.700	1.883
0.480	2.100	2.387
0.650	2.700	3.927
8.000	5.802	6.000

50

TIME	VRMS	VINT
0.080	1.450	1.450
0.200	1.700	1.848
0.350	2.450	3.186
0.770	3.900	4.024
8.000	5.801	6.000

171

TIME	VRMS	VINT
0.080	1.450	1.450
0.220	1.700	1.819
0.800	2.300	2.502
1.150	3.500	5.306
8.000	5.708	6.000

286

TIME	VRMS	VINT
0.100	1.450	1.450
0.220	1.700	1.883
0.580	2.300	2.599
0.860	2.950	3.971
8.000	5.750	6.000

435

TIME	VRMS	VINT
0.100	1.450	1.450
0.200	1.700	1.918
0.450	2.400	2.836
0.750	3.350	4.406
8.000	5.803	6.000

82

TIME	VRMS	VINT
0.080	1.450	1.450
0.220	1.750	1.900
0.820	2.500	2.724
1.200	2.900	3.553
8.000	5.832	6.000

204

TIME	VRMS	VINT
0.080	1.450	1.450
0.250	1.700	1.806
0.720	2.500	2.895
1.020	3.400	4.930
8.000	5.734	6.000

328

TIME	VRMS	VINT
0.100	1.450	1.450
0.300	1.800	1.952
0.550	2.300	2.655
0.820	2.700	3.865
8.000	5.750	6.000

115

TIME	VRMS	VINT
0.080	1.450	1.450
0.400	1.950	2.058
0.940	2.650	4.028
0.870	3.300	4.150
8.000	5.781	6.000

236

TIME	VRMS	VINT
0.100	1.450	1.450
0.260	1.650	1.751
0.550	2.550	3.228
1.150	3.400	4.024
8.000	5.700	6.000

358

TIME	VRMS	VINT
0.090	1.850	1.450
0.250	1.750	1.898
0.480	2.000	2.240
0.570	2.500	4.272
8.000	5.821	6.000

DURHAM UNIVERSITY

SEISMIC PROCESSING

FARNELLA 1/81 LINE : 4

SYSTEM PARAMETERS

×××FIELD DATA×××
COLLECTED ON FARNELLA CRUISE NO: 1/81
SOURCE/RECEIVER OFFSET 245 METRES
RECEIVER SPACING 100 METRES
SAMPLING INTERVAL 4 MSEC

×××DEMULTIPLEX×××

×××SORT×××
FOLD 24
COMMON MID-POINT SPACING 50 METRES
RECORDER ERROR REMOVAL

×××OBS×××
TIME BREAK CORRECTION 64 MSEC
SOURCE DEPTH 10 METRES
RECEIVER DEPTH 10 METRES
POLARITY REVERSAL TRACES 2,5,10 AND 12
BAND-PASS FILTER 5,10 TO 40,60 HZ
VELOCITY FILTER OPERATOR SIZE 11×11
FALSE VELOCITY STRUCTURE

TIME (MSEC)	RMS VELOCITY (KM/SEC)
400	1.4
1000	2.1
2000	3.0

MUTE

TRACE NUMBER	TIME (MSEC)
4	500
7	1100
24	2800

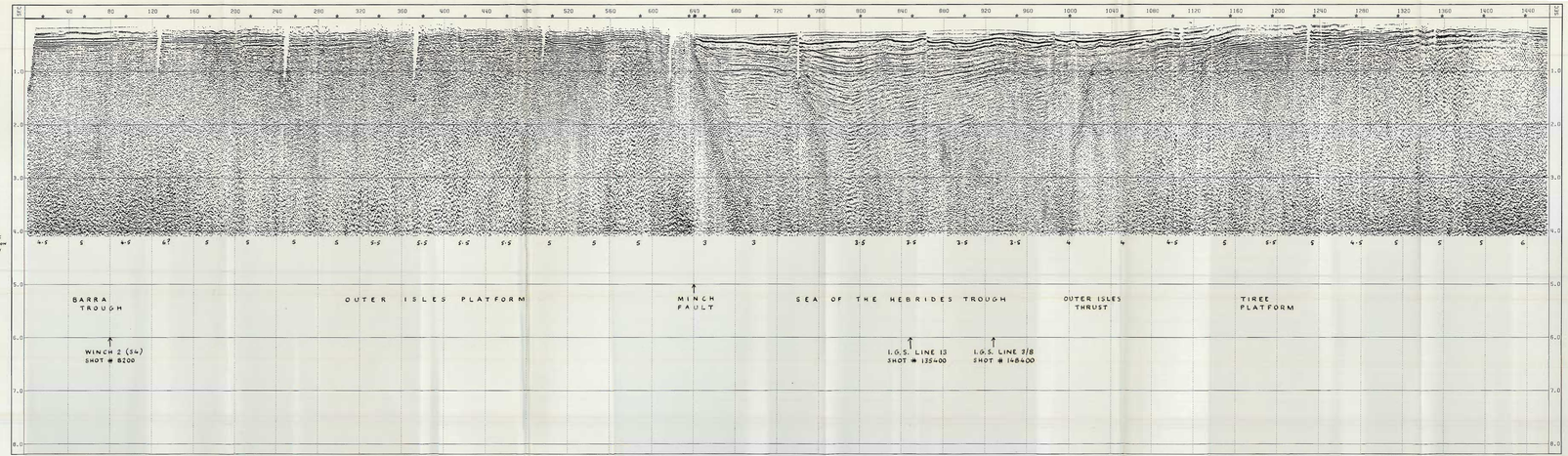
×××STACK×××
RESAMPLED PRIOR TO NMO 1 MSEC
NMO AND STACK 24 FOLD

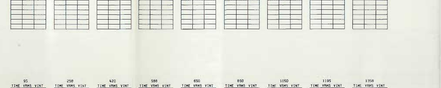
×××DAS×××
BAND-PASS FILTER 5,10 TO 40,60 HZ
GEOMETRIC SPREADING CORRECTION
DECONVOLUTION PREDICTION ERROR
AUTOCORR. DESIGN 2000 MSEC FROM 0 MSEC
ACTIVE LENGTH 400 MSEC
PREDICTION LAG 100 MSEC
PRE-WHITENING 1 %
TRACE NORMALIZATION UNIT ENERGY

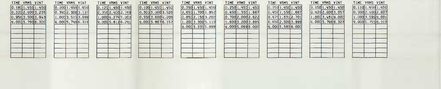
×××DISPLAY×××
GAIN 75
CLIP 0.005 TO 0.05 INCHES
TRACE SPACING 0.025 INCHES
VERTICAL SCALE 1.25 INCHES/SEC

PROCESSED BY R.W.HOBBS

FARNELLA 1/81 LINE : 7 STORE GATHER NO. 1 END GATHER NO. 1458
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.025 INCHES TRACE LENGTH 5.12 INCHES HIGH CLIP LEVEL 0.050 INCHES
PLOTTING DATA 7.5000-001 LOW CLIP LEVEL 0.005 INCHES







DURHAM UNIVERSITY
SEISMIC PROCESSING
FARNELLA 1/81 LINE : 7

SYSTEM PARAMETERS

FIELD DATA
COLLECTED ON FARNELLA CRUISE NO. 1/81
NOMINAL RECEIVER-OFFSET 245 METRES
RECEIVER INTERVAL 100 METRES
SAMPLING INTERVAL 4 MSEC

DEMULTIPLEX

SORT
FOLD 24
COMMON MID-POINT SPACING 50 METRES
RECORDER ERROR REMOVAL

DBS
TIME BREAK CORRECTION 64 MSEC
SOURCE DEPTH 10 METRES
RECEIVER DEPTH 10 METRES
POLARITY REVERSAL TRACES 2,5,10 AND 12
BAND-PASS FILTER 5.00 TO 60.00 HZ
VELOCITY FILTER PARAMETER SIZE 11*11
PAIRS VELOCITY STRUCTURE

TIME (MSEC)	RMS VELOCITY (KM/SEC)
400	1.4
1000	2.1
2000	3.0

MUTE

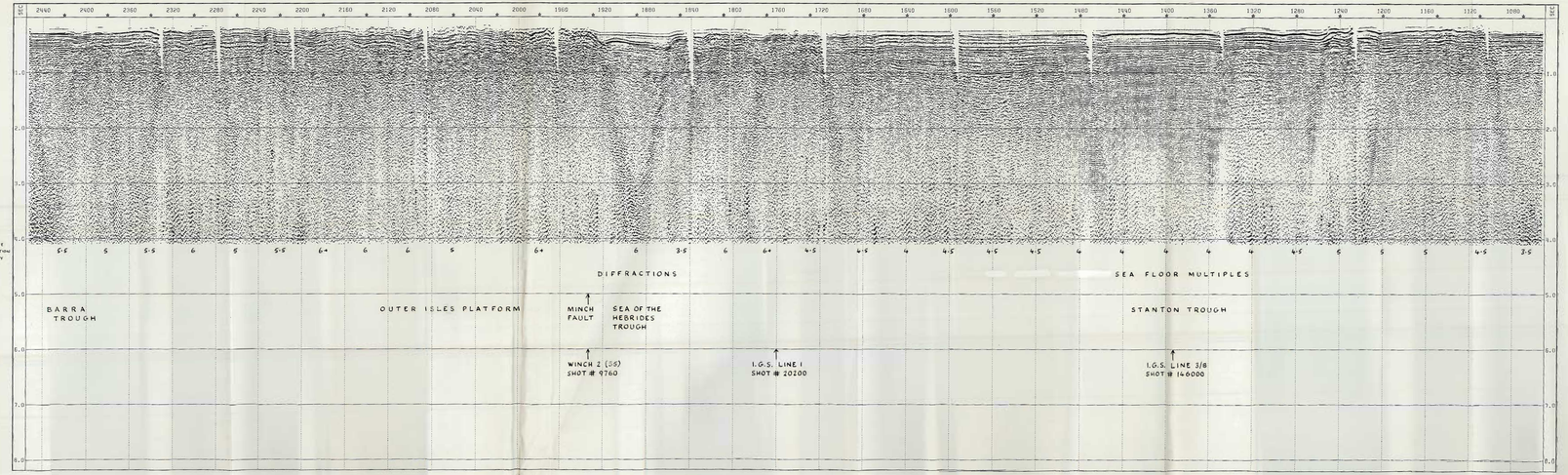
TRACE NUMBER	TIME (MSEC)
4	500
7	1100
24	2800

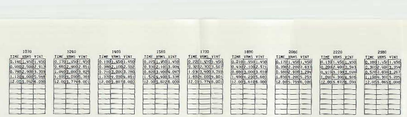
STACK
RESAMPLED PRIOR TO NMO 1 MSEC
NMO AND STACK 24 FOLD

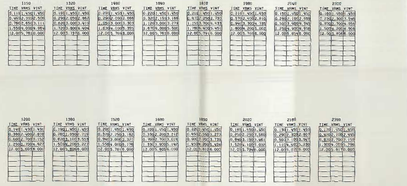
DBS
BAND-PASS FILTER 5.10 TO 40.60 HZ
GEOMETRIC SPREADING CORRECTION
DECONVOLUTION
PREDICTION ERROR
AUTOCORR. DESIGN 2000 MSEC FROM 0 MSEC
ACTIVE LENGTH 430 MSEC
PREDICTION LAG 100 MSEC
PRE-WHITENING 1 %
TRACE NORMALIZATION UNIT ENERGY

DISPLAY
GAIN 75
CLIP 0.005 TO 0.05 INCHES
TRACE SPACING 0.025 INCHES
VERTICAL SCALE 1.25 INCHES/SEC

PROCESSED BY R.W. HOBBS





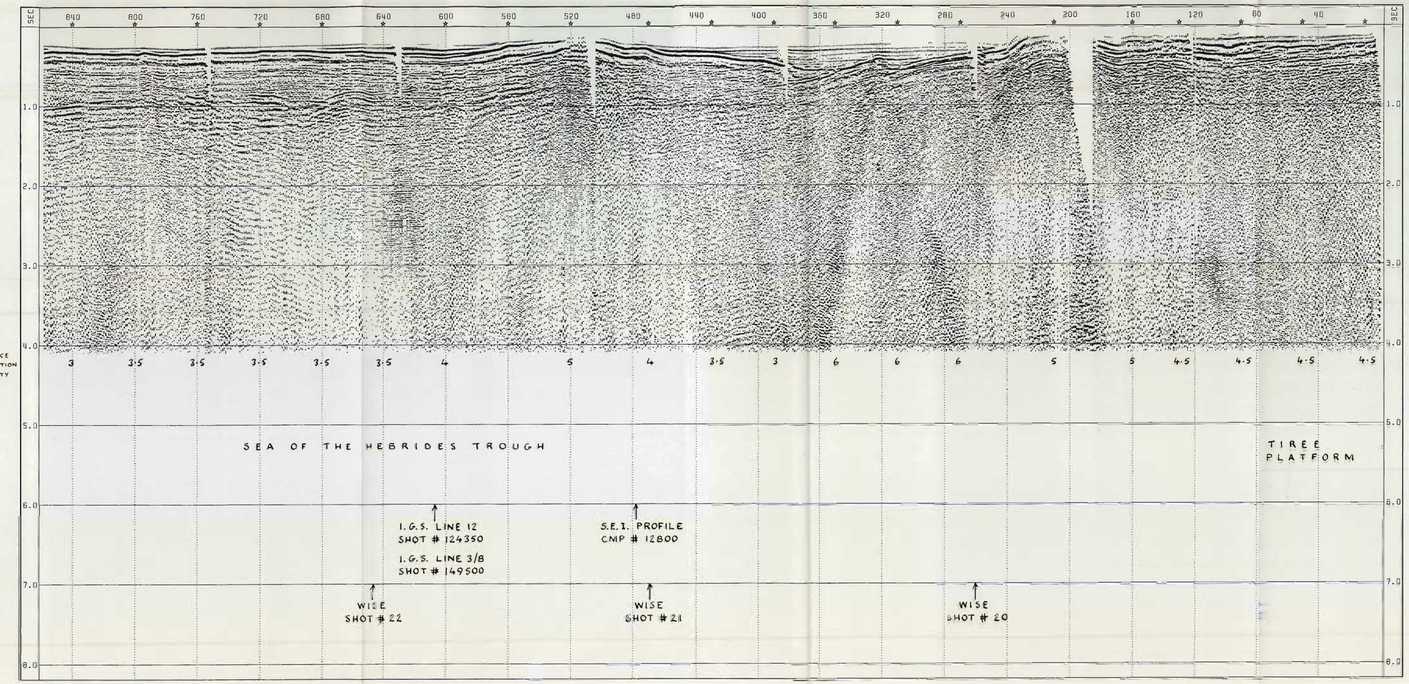


DURHAM UNIVERSITY
SEISMIC PROCESSING
FARNELLA 1/81 LINE : 6

SYSTEM PARAMETERS

FIELD DATA	
COLLECTED ON	FARNELLA CRUISE NO: 1/81
SOURCE/RECEIVER OFFSET	245 METRES
RECEIVER SPACING	100 METRES
SAMPLING INTERVAL	4 MSEC
DEMULTIPLEX	
SORT	
FOLD	24
COMMON MID-POINT SPACING	50 METRES
RECORDED TANOA REMOVAL	
DBS	
TIME BREAK CORRECTION	84 MSEC
SOURCE DEPTH	10 METRES
RECEIVER DEPTH	10 METRES
POLARITY REVERSAL TRACES	2,5,10 AND 12
BAND-PASS FILTER	5,10 TO 40,60 NZ
VELOCITY FILTER	OPERATOR SIZE 11×11
	TRACE VELOCITY STRUCTURE
	TIME (MSEC) RMS VELOCITY (KM/SEC)
	400 1.4
	1600 2.0
	2000 3.0
MUTE	TRACE NUMBER TIME (MSEC)
	5 500
	7 1100
	24 2800
STACK	
RESAMPLED PRIOR TO NMO AND AND STACK	1 MSEC 24 FOLD
DBS	
BAND-PASS FILTER	5,10 TO 40,50 NZ
GEOMETRIC SPREADING CORRECTION	
DECONVOLUTION	PREDICTION ERROR
	AUTOCORR. DESIGN 2000 MSEC FROM 0 MSEC
	ACTIVE LENGTH 400 MSEC
	PREDICTION LAG 100 MSEC
	PRE-WHITENING 1 %
TRACE NORMALIZATION	UNIT ENERGY
DISPLAY	
GAIN	25
CLIP	0.005 TO 0.05 INCHES
TRACE SPACING	0.025 INCHES
VERTICAL SCALE	1.25 INCHES/SEC
PROCESSED BY	H.W. MOBBS

FARNELLA 1/81 LINE : 8 START GATHER NO. 1 END GATHER NO. 858
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.025 INCHES TRACE LENGTH 5.12 INCHES HIGH CLIP LEVEL 0.050 INCHES
PLOTTING GAIN 7.500E+01 LOW CLIP LEVEL 0.005 INCHES

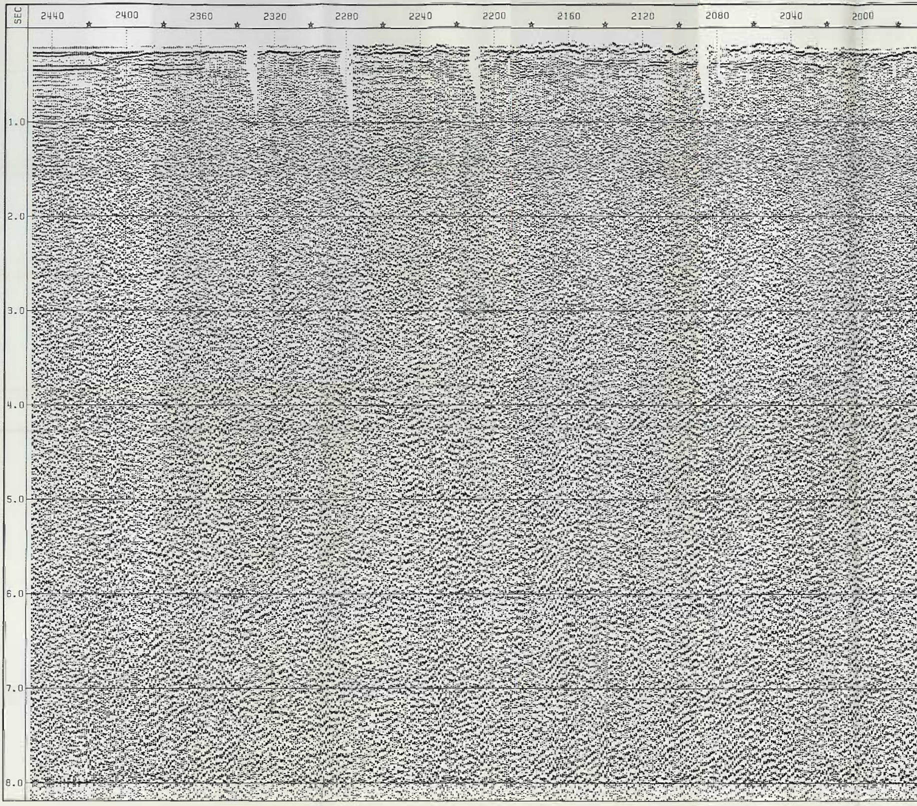


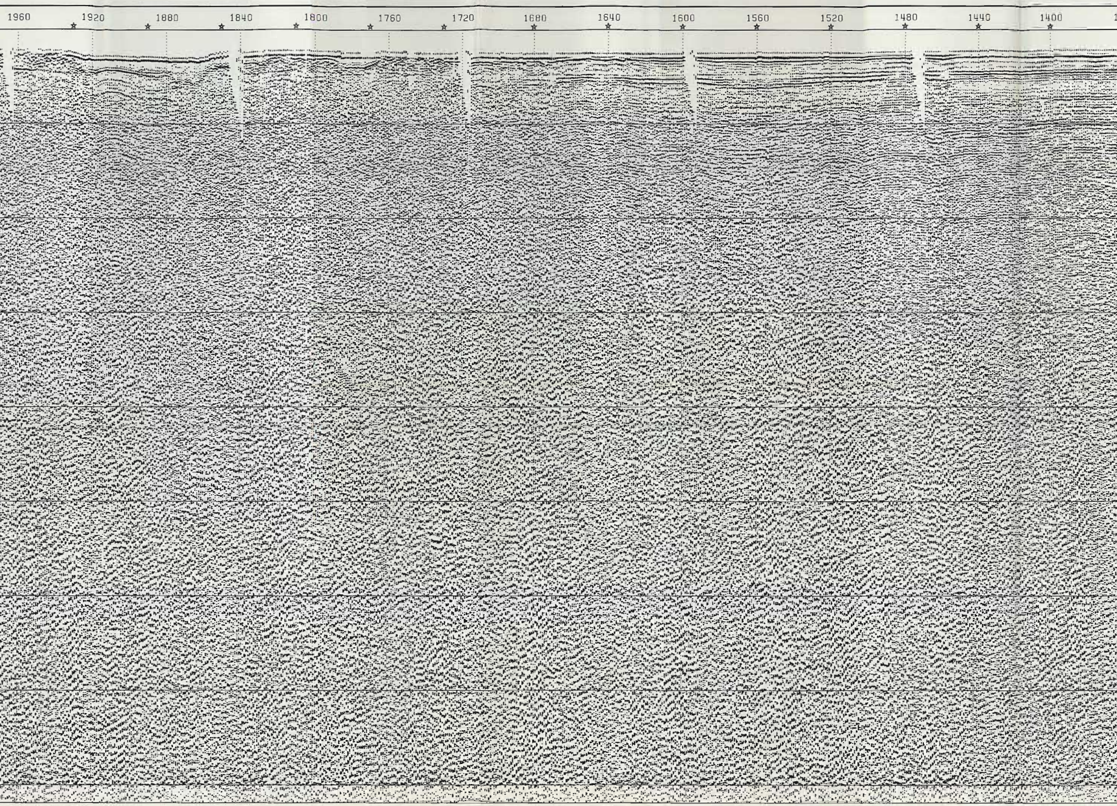
DURHAM UNIVERSITY
SEISMIC PROCESSING
FARNELLA 1/81 LINE : 8

SYSTEM PARAMETERS

×××FIELD DATA×××	
COLLECTED ON	FARNELLA CRUISE NO: 1/81
SOURCE/RECEIVER OFFSET	245 METRES
RECEIVER SPACING	100 METRES
SAMPLING INTERVAL	4 MSEC
×××DEMULTIPLEX×××	
×××SORT×××	
FOLD	24
COMMON MID-POINT SPACING	50 METRES
RECORDER ERROR REMOVAL	
×××DBS×××	
TIME BREAK CORRECTION	64 MSEC
SOURCE DEPTH	10 METRES
RECEIVER DEPTH	10 METRES
POLARITY REVERSAL TRACES	2,5,10 AND 12
BAND-PASS FILTER	5,10 TO 40,60 HZ
VELOCITY FILTER	OPERATOR SIZE 11×11
	FALSE VELOCITY STRUCTURE
	TIME (MSEC) RMS VELOCITY (KM/SEC)
	400 1.4
	1000 2.1
	2000 3.0
MUTE	TRACE NUMBER TIME (MSEC)
	4 500
	7 1100
	24 2800
×××STACK×××	
RESAMPLED PRIOR TO NMO	1 MSEC
NMO AND STACK	24 FOLD
×××DAS×××	
BAND-PASS FILTER	5,10 TO 40,60 HZ
GEOMETRIC SPREADING CORRECTION	
DECONVOLUTION	PREDICTION ERROR
	AUTOCORR. DESIGN 2000 MSEC FROM 0 MSEC
	ACTIVE LENGTH 400 MSEC
	PREDICTION LAG 100 MSEC
	PRE-WHITENING 1 %
TRACE NORMALIZATION	UNIT ENERGY
×××DISPLAY×××	
GAIN	75
CLIP	0.005 TO 0.05 INCHES
TRACE SPACING	0.025 INCHES
VERTICAL SCALE	1.25 INCHES/SEC
PROCESSED BY	R.W.HOBBS

FARNELLA 1/81 LINE : 6 START GATHER NO. 1051 END GATHER NO. 2450
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.025 INCHES TRACE LENGTH 10.24 INCHES 50 SAMPLE AGC APPLIED
PLOTTING GAIN 2.500E+00 LOW CLIP LEVEL 0.005 INCHES HIGH CLIP LEVEL 0.050 INCHES





1070

TIME	VRMS	VINT
0.140	1.450	1.450
0.400	2.500	2.913
0.740	2.900	3.309
1.120	4.000	5.548
12.00	5.842	6.000

1110

TIME	VRMS	VINT
0.140	1.450	1.450
0.290	2.300	2.875
0.770	3.100	3.496
1.550	4.100	4.891
12.00	5.790	6.000

1150

TIME	VRMS	VINT
0.110	1.450	1.450
0.460	2.300	2.506
0.760	2.650	3.111
1.550	4.000	4.964
12.00	5.781	6.000

1200

TIME	VRMS	VINT
0.140	1.450	1.450
0.380	2,400	2.610
0.600	2.700	3.152
1.250	3.700	4.427
12.00	5.803	6.000

1240

TIME	VRMS	VINT
0.170	1.450	1.450
0.660	2.400	2.651
1.040	3.000	3.825
1.600	4.000	5.385
12.00	5.774	6.001

1280

TIME	VRMS	VINT
0.190	1.450	1.450
0.290	2.000	2.758
0.940	3.350	3.801
1.600	4.200	5.175
12.00	5.792	6.000

1320

TIME	VRMS	VINT
0.190	1.450	1.450
0.290	2.050	2.862
0.820	3.000	3.410
1.720	3.800	4.404
12.00	5.737	6.000

1360

TIME	VRMS	VINT
0.190	1.450	1.450
0.280	1.950	2.719
0.820	3.100	3.553
1.500	4.200	5.227
12.00	5.806	6.000

1400

TIME	VRMS	VINT
0.190	1.450	1.450
0.380	2.100	2.592
0.710	3.000	3.780
1.370	4.000	4.651
12.00	5.807	6.000

1440

TIME	VRMS	VINT
0.200	1.450	1.450
0.410	2.300	2.885
0.780	3.000	3.621
1.280	3.700	4.583
12.00	5.798	6.000

1480

TIME	VRMS	VINT
0.200	1.450	1.450
0.290	2.000	2.866
1.050	3.000	3.303
1.630	4.000	5.355
12.00	5.769	6.000

1520

TIME	VRMS	VINT
0.200	1.450	1.450
0.320	2.250	3.162
0.940	3.000	3.321
1.550	4.000	5.176
12.00	5.781	6.000

1560

TIME	VRMS	VINT
0.220	1.450	1.450
0.330	2.100	3.004
0.820	3.400	4.047
1.520	4.400	5.338
12.00	5.822	6.000

1600

TIME	VRMS	VINT
0.220	1.450	1.450
0.320	2.100	3.080
1.050	3.200	3.577
1.500	4.000	5.426
12.00	5.788	6.000

1640

TIME	VRMS	VINT
0.220	1.450	1.450
0.320	2.150	3.188
1.120	3.000	3.279
1.670	4.000	5.500
12.00	5.783	6.000

1680

TIME	VRMS	VINT
0.220	1.450	1.450
0.330	2.200	3.212
0.780	2.700	3.014
1.330	3.900	5.142
12.00	5.805	6.000

1730

TIME	VRMS	VINT
0.220	1.450	1.450
0.320	2.300	3.507
1.030	3.400	3.793
1.600	4.000	4.901
12.00	5.774	6.001

1770

TIME	VRMS	VINT
0.220	1.450	1.450
0.340	2.600	3.911
0.950	3.300	3.532
1.180	4.000	6.092
12.00	5.834	6.000

1610

TIME	VRMS	VINT
0.210	1.450	1.450
0.470	2.250	2.730
1.150	3.700	4.433
1.780	4.400	5.451
12.00	5.791	6.000

1650

TIME	VRMS	VINT
0.220	1.450	1.450
0.450	2.550	3.273
0.900	3.200	3.739
1.450	4.200	5.454
12.00	5.612	6.000

1890

TIME	VRMS	VINT
0.210	1.450	1.450
0.430	2.100	2.571
0.880	3.000	3.659
1.400	4.200	5.680
12.00	5.819	6.000

1930

TIME	VRMS	VINT
0.210	1.450	1.450
0.520	2.250	2.659
0.800	3.300	4.660
1.450	4.200	5.094
12.00	5.812	6.000

1980

TIME	VRMS	VINT
0.210	1.450	1.450
0.570	2.400	2.810
0.940	3.300	4.335
1.600	4.200	5.002
12.00	5.766	6.000

2020

TIME	VRMS	VINT
0.180	1.450	1.450
0.250	2.250	3.560
0.840	3.150	3.461
1.520	4.100	5.032
12.00	5.794	6.000

2060

TIME	VRMS	VINT
0.170	1.450	1.450
0.390	2.200	2.697
0.960	2.900	3.294
1.850	4.200	5.253
12.00	5.759	6.000

2100

TIME	VRMS	VINT
0.160	1.450	1.450
0.270	2.400	3.329
0.770	3.000	3.279
1.570	4.100	4.932
12.00	5.767	6.000

2140

TIME	VRMS	VINT
0.150	1.450	1.450
0.280	2.100	2.659
0.500	3.400	4.545
1.250	4.400	4.956
12.00	5.854	6.000

2180

TIME	VRMS	VINT
0.130	1.450	1.450
0.290	2.300	2.807
0.500	3.100	3.947
1.110	4.400	5.230
12.00	5.870	6.000

2220

TIME	VRMS	VINT
0.130	1.450	1.450
0.200	2.400	3.543
0.410	3.100	3.644
1.040	4.300	4.926
12.00	5.872	6.000

2260

TIME	VRMS	VINT
0.150	1.450	1.450
0.360	2.500	3.035
0.950	3.700	4.270
1.500	4.200	4.946
12.00	5.806	6.000

2300

TIME	VRMS	VINT
0.160	1.450	1.450
0.250	2.300	3.546
0.930	3.700	4.056
1.500	4.200	4.908
12.00	5.806	6.000

2340

TIME	VRMS	VINT
0.170	1.450	1.450
0.420	2.100	2.445
0.870	2.700	3.159
1.300	4.000	5.798
12.00	5.617	6.000

2380

TIME	VRMS	VINT
0.180	1.450	1.450
0.300	2.400	3.354
0.520	2.800	3.267
1.100	4.300	5.295
12.00	5.665	6.000

2420

TIME	VRMS	VINT
0.180	1.450	1.450
0.330	2.500	3.351
0.600	2.900	3.324
1.600	4.000	4.534
12.00	5.774	6.001

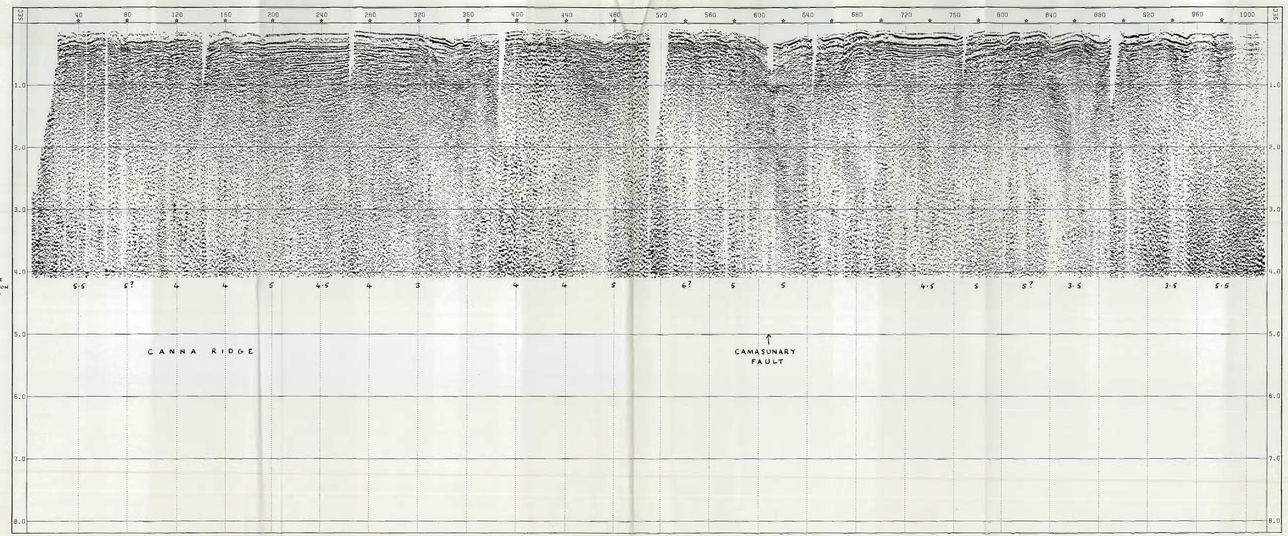
DURHAM UNIVERSITY
SEISMIC PROCESSING
FARNELLA 1/81 LINE : 6

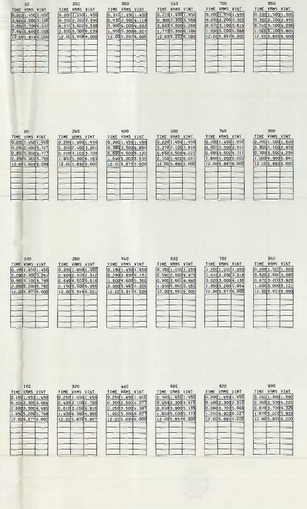
SYSTEM PARAMETERS

```
***FIELD DATA***
COLLECTED ON               FARNELLA CRUISE NO: 1/81
SOURCE/RECEIVER OFFSET     245 METRES
RECEIVER SPACING           100 METRES
SAMPLING INTERVAL          4 MSEC

***DEMULTIPLEX***

***SORT***
FOLD                       24
COMMON MID-POINT SPACING   50 METRES
RECORDER ERROR REMOVAL

***DBS***
TIME BREAK CORRECTION      64 MSEC
SOURCE DEPTH               10 METRES
RECEIVER DEPTH             10 METRES
POLARITY REVERSAL TRACES   2,5,10 AND 12
BAND-PASS FILTER           5,10 TO 40,60 HZ
TIME RAMP                  EXP( T )
DECONVOLUTION              PREDICTION ERROR
                           AUTOCORR. DESIGN       4000 MSEC FROM 0 MSEC
                           ACTIVE LENGTH           400 MSEC
                           PREDICTION LAG           40 MSEC
                           PRE-WHITENING             1 %
REMOVE TIME RAMP
MUTE                       TRACE NUMBER    TIME (MSEC)
                                2             500
                               24            2700

***STACK***
RESAMPLED PRIOR TO NMO     2 MSEC
NMO AND STACK              24 FOLD

***DAS***
BAND-PASS FILTER           5,10 TO 40,60 HZ
GEOMETRIC SPEADING CORRECTION
DECONVOLUTION              PREDICTION ERROR
                           AUTOCORR. DESIGN       2000 MSEC FROM 0 MSEC
                           ACTIVE LENGTH           400 MSEC
                           PREDICTION LAG          100 MSEC
                           PRE-WHITENING             1 %
TRACE NORMALIZATION        UNIT ENERGY

***DISPLAY***
A.G.C. APPLIED             200 MSEC WINDOW
GAIN                       2.5
CLIP                       0.005 TO 0.05 INCHES
TRACE SPACING              0.025 INCHES
VERTICAL SCALE             1.25 INCHES/SEC

PROCESSED BY               R.W.HOBBS
```

FARNELLA 1/81 LINE : 9 START GATHER NO. 1 END GATHER NO. 1014
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.025 INCHES TRACE LENGTH 5.12 INCHES HIGH CLIP LEVEL 0.050 INCHES
PLOTTING GRIN 7.50E+01 LOW CLIP LEVEL 0.005 INCHES

SURFACE REFRACTION VELOCITY KM/S





DURHAM UNIVERSITY

SEISMIC PROCESSING

FARNELLA 1/81 LINE : 9

SYSTEM PARAMETERS

FIELD DATA

COLLECTED ON	FARNELLA CRUISE NO: 1/81
SOURCE/RECEIVER OFFSET	245 METRES
RECEIVER SPACING	100 METRES
SAMPLING INTERVAL	4 MSEC

DEMULTIPLEX

SORT

FOLD	24
COMMON MID-POINT SPACING	50 METRES
RECORDER ERROR REMOVAL	

DBS

TIME BREAK CORRECTION	64 MSEC
SOURCE DEPTH	10 METRES
RECEIVER DEPTH	10 METRES
POLARITY REVERSAL TRACES	1,2,5,10,12 AND 24
BAND-PASS FILTER	5.10 TO 40.60 HZ
VELOCITY FILTER	(OPERATOR SIZE 11*11
	FALSE VELOCITY STRUCTURE

TIME (MSEC)	RMS VELOCITY (KM/SEC)
400	1.4
1000	2.1
2000	3.0

MUTE	TRACE NUMBER	TIME (MSEC)
	4	500
	7	1100
	24	2800

STACK

RESAMPLED PRIOR TO NMO	1 MSEC
NMO AND STACK	24 FOLD

DBS

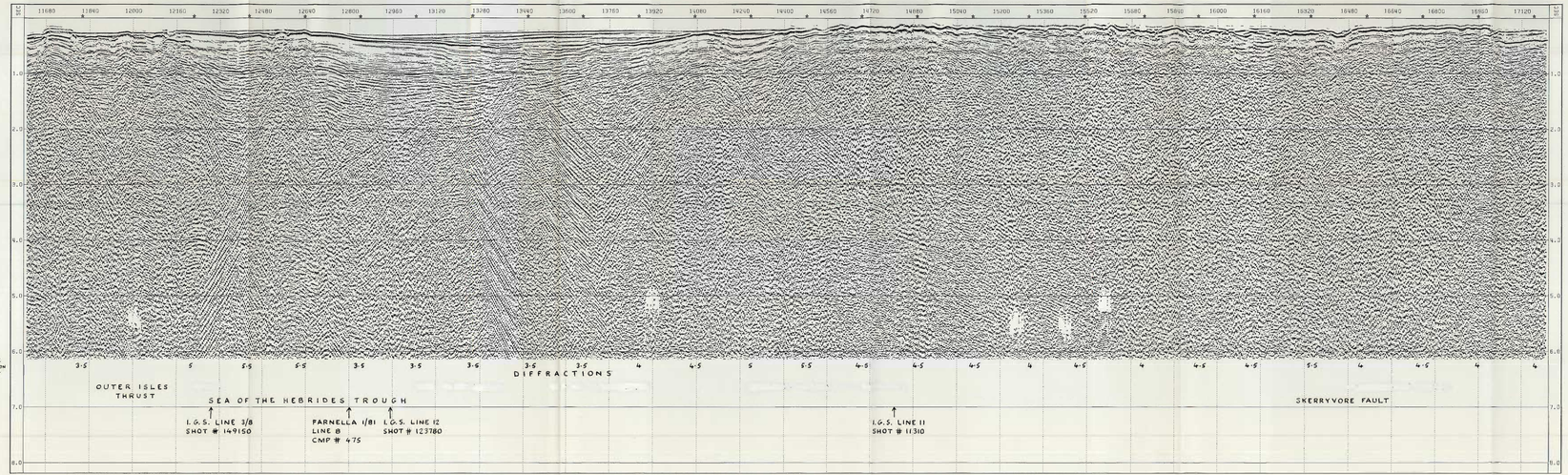
BAND-PASS FILTER	5.10 TO 40.60 HZ
GEOMETRIC SPREADING CORRECTION	
DECONVOLUTION	PREDICTION ERROR
	AUTOCORR. DESIGN 2000 MSEC FROM 0 MSEC
	ACTIVE LENGTH 400 MSEC
	PREDICTION LAG 100 MSEC
	PRE-WHITENING 1 %
TRACE NORMALIZATION	UNIT ENERGY

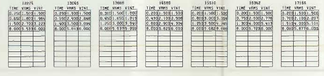
DISPLAY

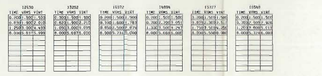
GRIN	75
CLIP	0.005 TO 0.05 INCHES
TRACE SPACING	0.025 INCHES
VERTICAL SCALE	1.28 INCHES/SEC
PROCESSED BY	A.W.MOBBS

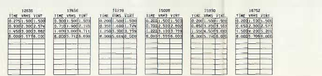
SEIS.EXP.INT. 1973 LINE = 11 START GATHER NO. 11612 END GATHER NO. 17216
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.025 INCHES TRACE LENGTH 7.68 INCHES 50 SAMPLE AGC APPLIED
PLOTTING GAIN 2.500E+00 LOW CLIP LEVEL 0.005 INCHES HIGH CLIP LEVEL 0.050 INCHES

SURFACE REFRACTION VELOCITY KM/S









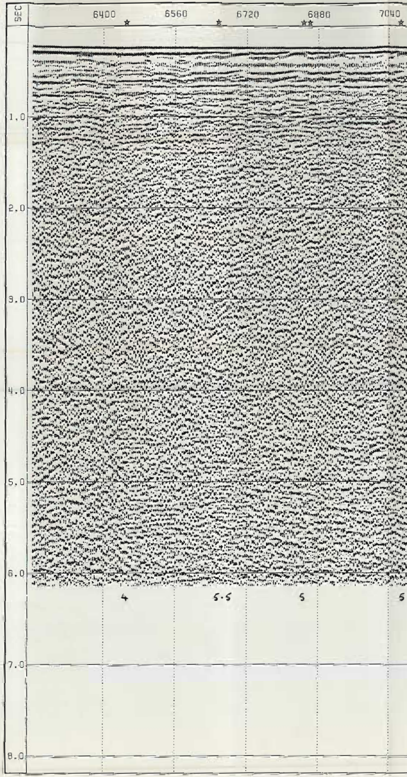
DURHAM UNIVERSITY
SEISMIC PROCESSING
SEIS.EXP.INT. 1973 LINE : 11

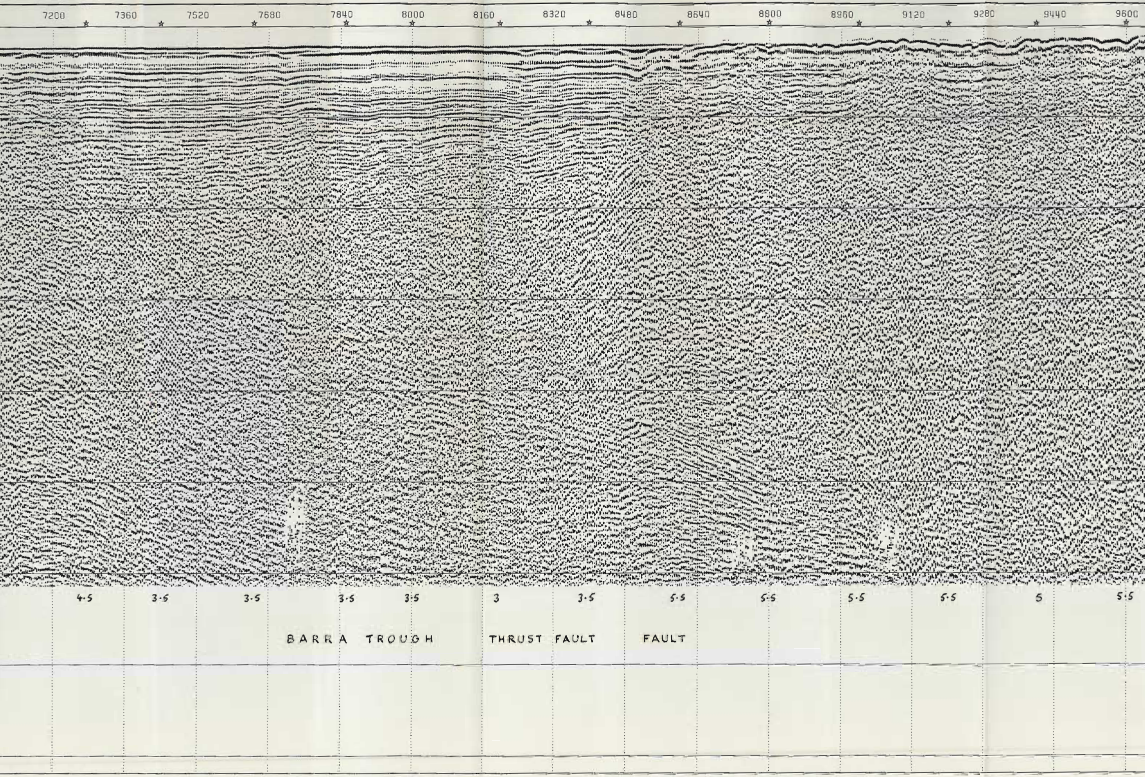
SYSTEM PARAMETERS

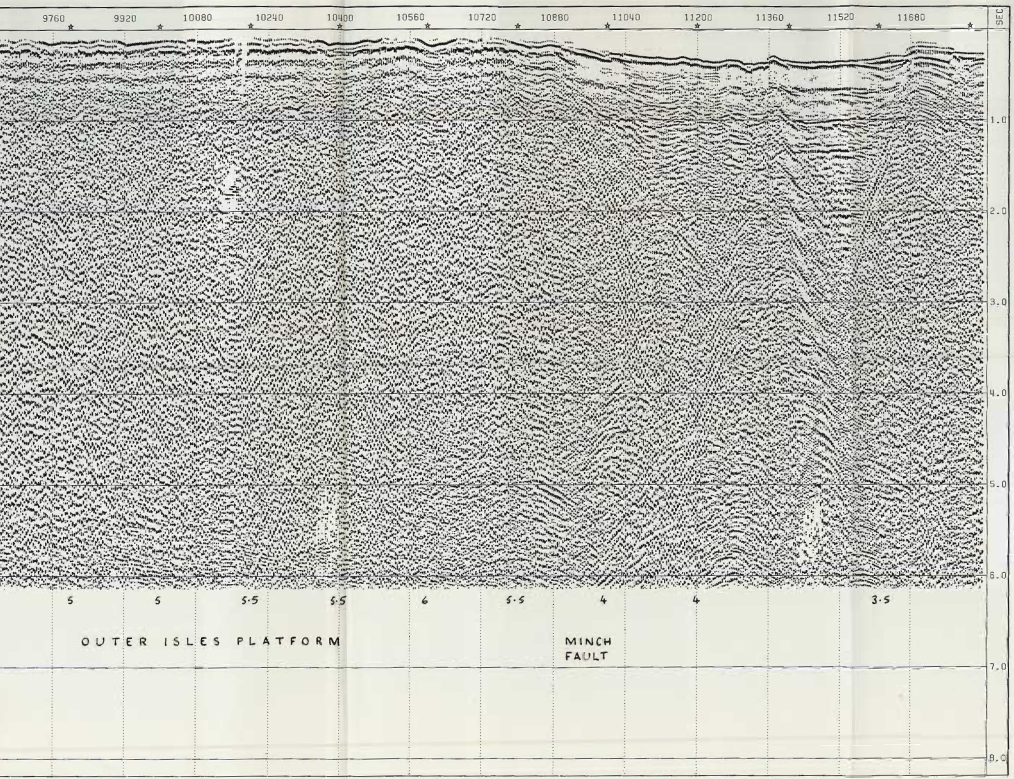
FIELD DATA	
COLLECTED BY	SEISMIC EXPLORATIONS INC.
YEAR	1973
SHOT SPACING	25 METRES
SOURCE/RECEIVER OFFSET	250 METRES
RECEIVER SPACING	25 METRES
SAMPLING INTERVAL	4 MSEC
DEMULTIPLEX	
SORT	
FOLD	24
COMMON MID-POINT SPACING	12.5 METRES
DSS	
KIND	5.10 TO 90.80 HZ
TIME RAMP	EXPONENTIAL
DECONVOLUTION	PREDICTION ERROR
	AUTOCORR. DESIGN 4000 MSEC FROM 0 MSEC
	PREDICTION LENGTH 400 MSEC
	PREDICTION LAG 40 MSEC
	PRE-WHITENING 1 %
REMOVE TIME RAMP	
MUTE	TRACE NUMBER TIME (MSEC)
	2 0
	24 2200
STACK	
RESAMPLED PRIOR TO NMO	2 MSEC
NMO AND STACK	24 FOLD
DSS	
TIME RAMP	T**2
DECONVOLUTION	PREDICTION ERROR
	AUTOCORR. DESIGN 2000 MSEC FROM 0 MSEC
	PREDICTION LENGTH 400 MSEC
	PREDICTION LAG 100 MSEC
	PRE-WHITENING 1 %
HORIZONTAL TRACE SUM	CMP SPACING 50 METRES
	EFFECTIVE FOLD 96
TRACE NORMALIZATION	UNIT ENERGY
DISPLAY	
A.G.C. APPLIED	200 MSEC WINDOW
GAIN	2.5
CLIP	0.005 TO 0.05
TRACE SPACING	0.025 INCHES
VERTICAL SCALE	1.25 INCHES/SEC
PROCESSED BY	R.W.HOBBS

SEIS.EXP.INT. 1973   LINE : 11   START GATHER NØ. 6244   END GATHER NØ. 11843
CØMMØN MID-PØINT   SECTIØN PLØT   PØSITIVE LØBE SHADED
TRACE SPACING 0.025 INCHES   TRACE LENGTH 7.68 INCHES   50 SAMPLE AGC APPLIED
PLØTTING GAIN 2.500E+00   LØW CLIP LEVEL 0.005 INCHES   HIGH CLIP LEVEL 0.050 INCHES

SURFACE
REFRACTION
VELOCITY
KM/S







6452

TIME	VRMS	VINT
0.210	1.500	1.500
0.400	2.100	2.607
1.400	3.700	4.172
8.000	5.665	6.000

6658

TIME	VRMS	VINT
0.200	1.500	1.500
0.600	2.500	2.872
1.850	4.500	5.315
8.000	5.723	6.000

6848

TIME	VRMS	VINT
0.200	1.500	1.500
0.600	1.900	2.071
1.300	3.000	3.691
8.000	5.622	5.999

6862

TIME	VRMS	VINT
0.200	1.500	1.500
0.500	2.000	2.273
1.500	4.500	5.327
8.000	5.749	5.000

Fig 7.9a

7068

TIME	VRMS	VINT
0.250	1.500	1.500
0.500	1.900	2.229
1.300	3.100	3.655
8.000	5.631	6.000

7852

TIME	VRMS	VINT
0.200	1.500	1.500
0.750	2.500	2.778
0.950	3.200	5.020
8.000	5.738	6.000

8600

TIME	VRMS	VINT
0.200	1.500	1.500
0.350	1.700	1.935
0.620	2.600	3.432
8.000	5.808	6.000

9400

TIME	VRMS	VINT
0.170	1.500	1.500
0.550	2.500	2.835
1.850	5.000	5.739
8.000	5.764	6.000

10200

TIME	VRMS	VINT
0.150	1.500	1.500
0.420	3.000	3.571
1.750	5.500	6.080
8.000	5.894	6.000

11000

TIME	VRMS	VINT
0.250	1.500	1.500
0.980	2.200	2.393
1.600	3.200	4.333
8.000	5.554	6.000

11814

TIME	VRMS	VINT
0.350	1.500	1.500
0.650	1.800	2.096
0.890	2.300	3.293
8.000	5.708	6.000

7274

TIME	VRMS	VINT
0.250	1.500	1.500
0.700	1.900	2.089
0.950	2.750	4.315
8.000	5.712	6.000

8000

TIME	VRMS	VINT
0.250	1.500	1.500
0.600	1.700	1.830
0.950	2.800	4.041
8.000	5.715	6.000

8800

TIME	VRMS	VINT
0.250	1.500	1.500
0.500	1.850	2.144
0.730	2.600	3.744
8.000	5.773	6.000

9600

TIME	VRMS	VINT
0.150	1.500	1.500
0.570	3.000	3.378
1.230	5.000	6.230
8.000	5.857	6.000

10400

TIME	VRMS	VINT
0.250	1.500	1.500
0.580	2.600	3.190
1.700	5.000	5.869
8.000	5.802	6.000

11200

TIME	VRMS	VINT
0.300	1.500	1.500
0.520	1.850	2.241
0.780	2.300	3.004
8.000	5.745	6.000

7440

TIME	VRMS	VINT
0.226	1.500	1.500
0.450	1.900	2.216
0.850	2.900	3.716
8.000	5.751	6.001

8200

TIME	VRMS	VINT
0.250	1.500	1.500
0.550	1.800	2.016
0.750	2.300	3.308
8.000	5.755	6.000

9000

TIME	VRMS	VINT
0.200	1.500	1.500
0.500	2.300	2.705
0.840	3.800	5.282
8.000	5.808	6.000

9800

TIME	VRMS	VINT
0.200	1.500	1.500
0.450	3.500	4.500
1.950	5.500	5.971
8.000	5.682	6.000

10600

TIME	VRMS	VINT
0.200	1.500	1.500
0.550	2.200	2.514
1.700	5.000	5.886
8.000	5.802	6.000

11406

TIME	VRMS	VINT
0.350	1.500	1.500
0.720	1.900	2.213
1.050	2.300	2.993
8.000	5.654	6.000

7648

TIME	VRMS	VINT
0.150	1.500	1.500
0.500	1.900	2.048
1.000	2.900	3.635
8.000	5.705	6.000

8400

TIME	VRMS	VINT
0.200	1.500	1.500
0.450	1.750	1.927
0.750	2.400	3.131
8.000	5.759	6.000

9200

TIME	VRMS	VINT
0.200	1.500	1.500
0.570	2.400	2.767
1.750	4.700	5.475
8.000	5.741	6.000

10060

TIME	VRMS	VINT
0.150	1.500	1.500
0.740	4.000	4.415
2.950	5.600	6.042
8.000	5.856	6.000

10800

TIME	VRMS	VINT
0.200	1.500	1.500
0.450	2.400	2.969
1.800	4.000	4.385
8.000	5.612	5.999

11610

TIME	VRMS	VINT
0.350	1.500	1.500
0.510	1.700	2.071
0.940	2.300	2.852
8.000	5.691	6.000

DURHAM UNIVERSITY

SEISMIC PROCESSING

SEIS.EXP.INT. 1973   LINE : 11

SYSTEM PARAMETERS

```
***FIELD DATA***
COLLECTED BY              SEISMIC EXPLORATIONS INC.
YEAR                      1973
SHOT SPACING              25 METRES
SOURCE/RECEIVER OFFSET    250 METRES
RECEIVER SPACING          25 METRES
SAMPLING INTERVAL         4 MSEC

***DEMULTIPLEX***

***SORT***
FOLD                      24
COMMON MID-POINT SPACING  12.5 METRES

***DBS***
BAND-PASS                 5,10 TO 30,80 HZ
TIME RAMP                 EXP(0.7*T)
DECONVOLUTION             PREDICTION ERROR
                          AUTOCORR. DESIGN      4000 MSEC FROM 0 MSEC
                          ACTIVE LENGTH          400 MSEC
                          PREDICTION LAG          40 MSEC
                          PRE-WHITENING            1 %
REMOVE TIME RAMP
MUTE                      TRACE NUMBER    TIME (MSEC)
                                2               0
                               24            2200

***STACK***
RESAMPLED PRIOR TO NMO    2 MSEC
NMO AND STACK             24 FOLD

***DAS***
TIME RAMP                 T(V**2)
DECONVOLUTION             PREDICTION ERROR
                          AUTOCORR. DESIGN      2000 MSEC FROM 0 MSEC
                          ACTIVE LENGTH          400 MSEC
                          PREDICTION LAG         100 MSEC
                          PRE-WHITENING            1 %
HORIZONTAL TRACE SUM      CMP SPACING             50 METRES
                          EFFECTIVE FOLD          96
TRACE NORMALIZATION       UNIT ENERGY

***DISPLAY***
A.G.C. APPLIED            200 MSEC WINDOW
GAIN                      2.5
CLIP                      0.005 TO 0.05
TRACE SPACING             0.025 INCHES
VERTICAL SCALE            1.25 INCHES/SEC

PROCESSED BY              R.W.HOBBS
```

FARNELLA 1/81 LINE : 6 START GATHER NO. 1 END GATHER NO. 1400
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.025 INCHES TRACE LENGTH 10.24 INCHES 50 SAMPLE AGC APPLIED
PLOTTING GAIN 2.5DB+40 LOW CLIP LEVEL 0.005 INCHES HIGH CLIP LEVEL 0.050 INCHES

DURHAM UNIVERSITY
SEISMIC PROCESSING
FARNELLA 1/81 LINE : 6

SYSTEM PARAMETERS

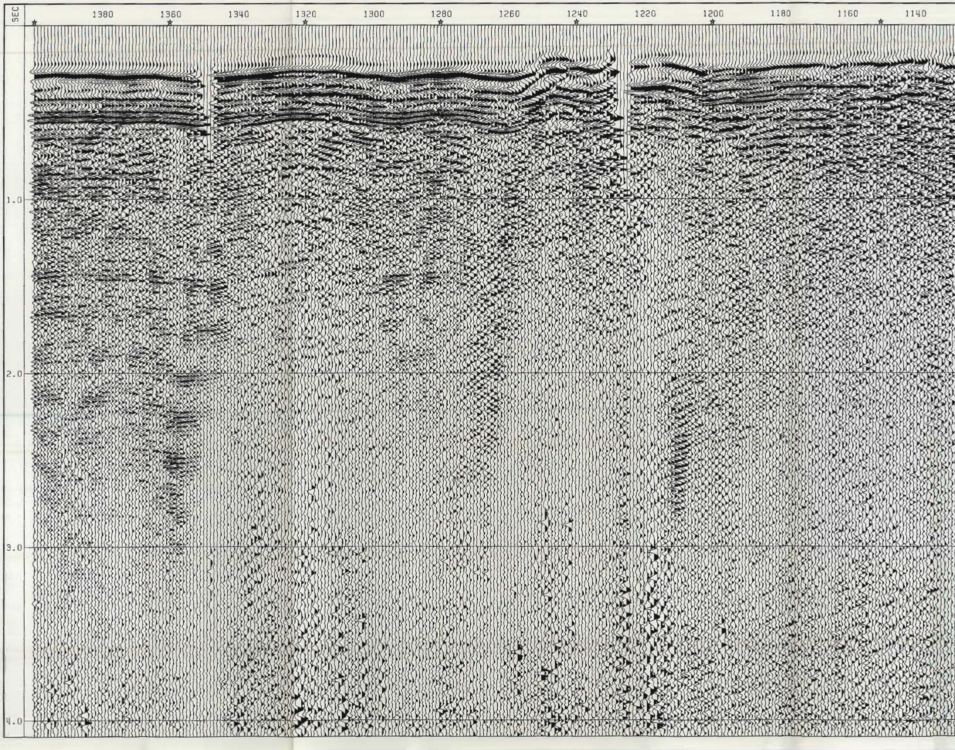
FIELD DATA		
COLLECTED ON	FARNELLA CRUISE NO: 1/81	
SOURCE/RECEIVER OFFSET	245 METRES	
RECEIVER SPACING	100 METRES	
SAMPLING INTERVAL	4 MSEC	
DEMULTIPLEX		
SORT		
FOLD	24	
COMMON MID-POINT SPACING	50 METRES	
RECORDER ERROR REMOVAL		
DBS		
TIME BREAK CORRECTION	84 MSEC	
SOURCE DEPTH	10 METRES	
RECEIVER DEPTH	10 METRES	
POLARITY REVERSAL TRACES	2,5,10,AND 12	
BAND-PASS FILTER	5,10 TO 40,60 HZ	
TIME RAMP	EXP: 1	
DECONVOLUTION	PREDICTION ERROR	
	AUTOCORR. DESIGN	4000 MSEC FROM 0 MSEC
	ACTIVE LENGTH	400 MSEC
	PREDICTION LAG	40 MSEC
	PRE-WHITENING	1 %
REMOVE TIME RAMP		
MUTE	TRACE NUMBER / TIME (MSEC)	
	2 / 500	
	24 / 2700	
STACK		
RESAMPLED PRIOR TO NMO	2 MSEC	
NMO AND STACK	24 FOLD	
DAS		
BAND-PASS FILTER	5,10 TO 40,60 HZ	
GEOMETRIC SPREADING CORRECTION		
DECONVOLUTION	PREDICTION ERROR	
	AUTOCORR. DESIGN	2000 MSEC FROM 0 MSEC
	ACTIVE LENGTH	400 MSEC
	PREDICTION LAG	100 MSEC
	PRE-WHITENING	1 %
TRACE NORMALIZATION	UNIT ENERGY	
DISPLAY		
A.G.C. APPLIED	200 MSEC WINDOW	
GAIN	2.5	
CLIP	0.005 TO 0.05 INCHES	
TRACE SPACING	0.025 INCHES	
VERTICAL SCALE	1.25 INCHES/SEC	
PROCESSED BY	A.W.HOBBS	

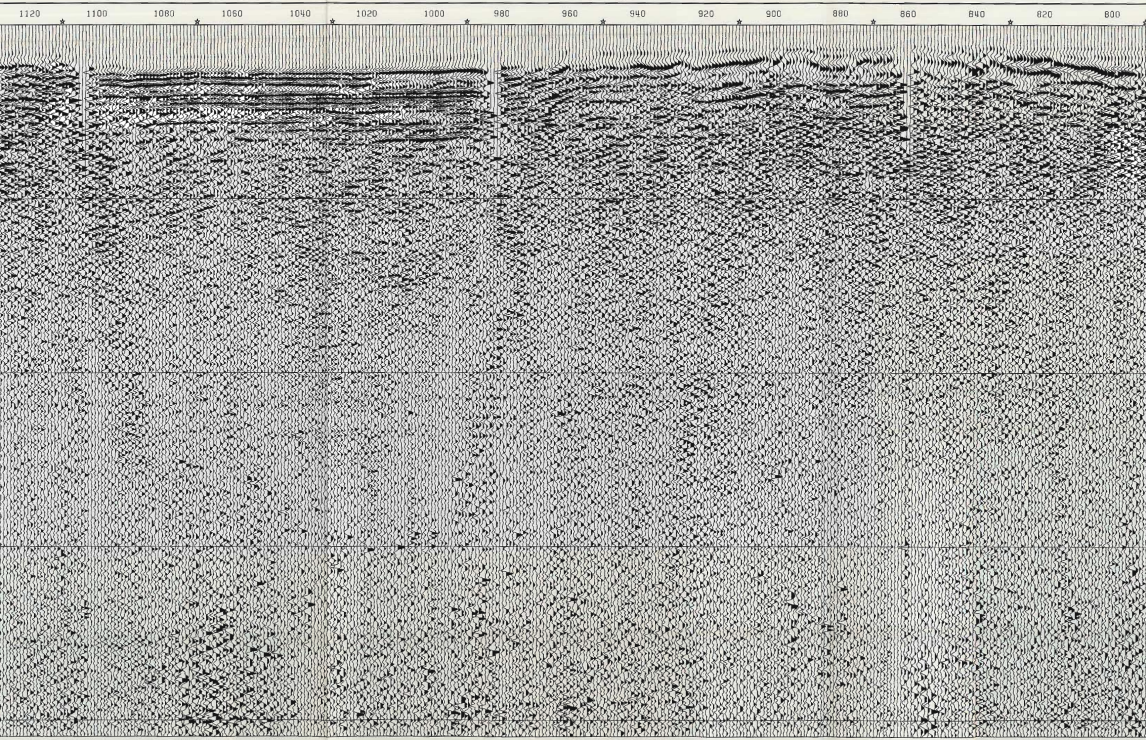
FARNELLA 1/81 LINE : 6 START GATHER NO. 1 END GATHER NO. 1400

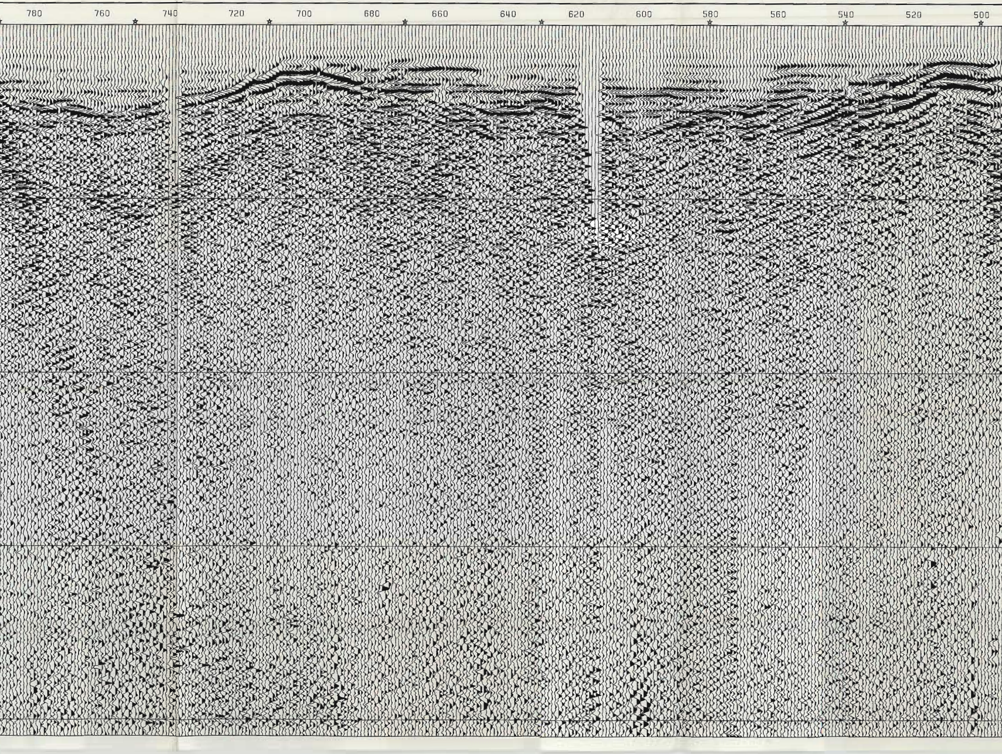
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED

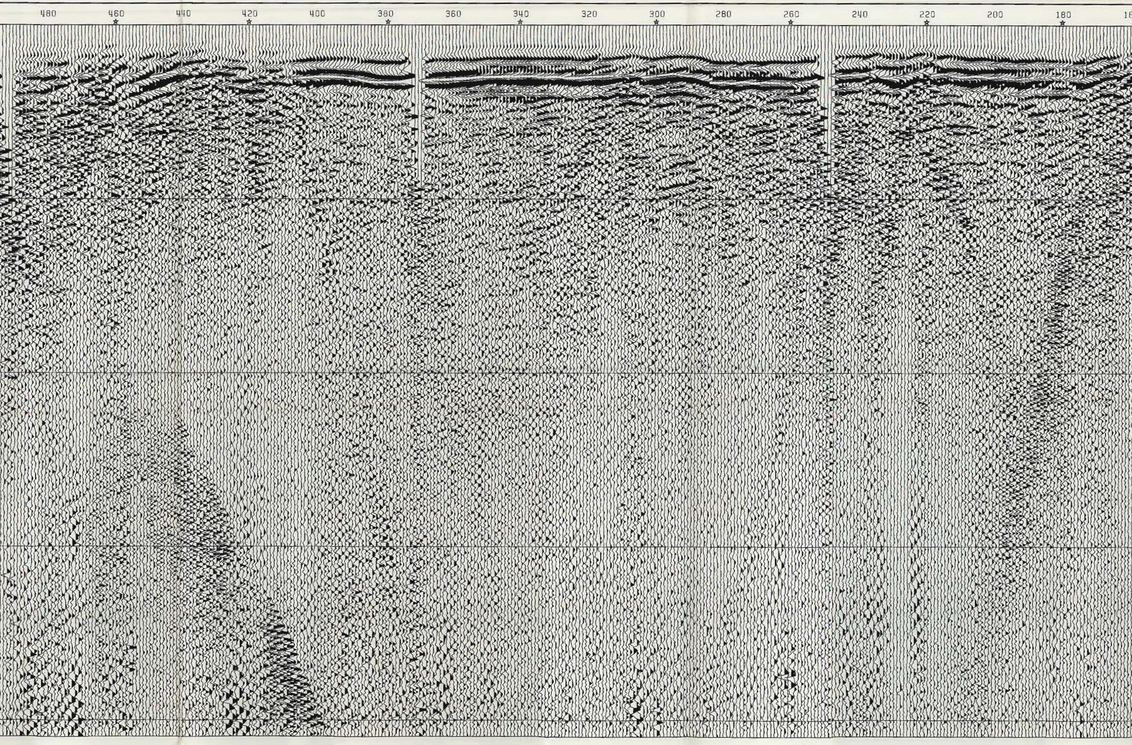
TRACE SPACING 0.050 INCHES TRACE LENGTH 10.24 INCHES

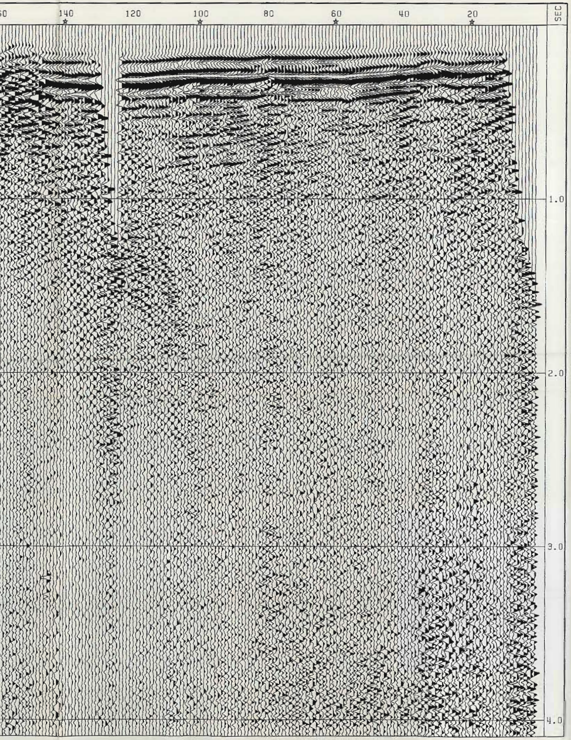
PLOTTING GAIN 1.500E+02 LOW CLIP LEVEL -0.100 INCHES HIGH CLIP LEVEL 0.100 INCHES











20

TIME	VRMS	VINT
0.250	1.450	1.450
0.510	1.650	1.822
0.760	2.350	3.288
1.400	2.800	3.280
12.00	5.720	6.000

180

TIME	VRMS	VINT
0.300	1.450	1.450
0.580	2.000	2.456
0.900	3.000	4.250
1.600	4.000	5.000
12.00	5.774	6.001

340

TIME	VRMS	VINT
0.290	1.450	1.450
0.580	2.150	2.673
1.010	2.800	3.490
1.280	3.800	6.255
12.00	5.805	6.000

500

TIME	VRMS	VINT
0.250	1.450	1.450
0.780	2.250	2.541
0.970	3.000	5.018
1.870	4.300	5.359
12.00	5.768	6.000

670

TIME	VRMS	VINT
0.200	1.450	1.450
0.480	1.950	2.240
0.920	3.300	4.315
1.350	4.100	5.429
12.00	5.817	5.000

830

TIME	VRMS	VINT
0.200	1.450	1.450
0.500	2.000	2.295
0.900	3.500	4.750
1.350	4.500	6.021
12.00	5.850	5.999

990

TIME	VRMS	VINT
0.170	1.450	1.450
0.550	2.000	2.202
1.050	3.500	4.618
1.750	4.500	5.679
12.00	5.805	6.000

1150

TIME	VRMS	VINT
0.110	1.450	1.450
0.460	2.300	2.508
0.760	2.850	3.111
1.550	4.000	4.964
12.00	5.781	6.000

60

TIME	VRMS	VINT
0.300	1.450	1.450
0.610	2.100	2.577
1.200	2.450	2.766
1.750	4.000	6.149
12.00	5.752	6.000

220

TIME	VRMS	VINT
0.200	1.450	1.450
0.400	1.750	2.006
0.980	2.400	2.760
1.220	3.400	5.937
12.00	5.769	6.000

379

TIME	VRMS	VINT
0.220	1.450	1.450
0.600	2.000	2.258
1.400	3.000	3.571
2.150	4.400	6.221
12.00	5.746	6.000

540

TIME	VRMS	VINT
0.220	1.450	1.450
0.560	2.100	2.430
0.920	2.700	3.431
1.600	4.600	6.319
12.00	5.833	6.000

710

TIME	VRMS	VINT
0.210	1.450	1.450
0.500	2.400	2.900
1.070	3.500	4.236
1.420	4.100	5.545
12.00	5.806	6.000

870

TIME	VRMS	VINT
0.170	1.450	1.450
0.500	2.300	2.833
0.790	2.950	3.819
1.150	4.100	5.862
12.00	5.845	6.000

1030

TIME	VRMS	VINT
0.140	1.450	1.450
0.370	2.600	3.096
0.690	3.000	3.404
1.280	4.000	4.918
12.00	5.820	6.001

1200

TIME	VRMS	VINT
0.140	1.450	1.450
0.380	2.400	2.810
0.600	2.700	3.152
1.250	3.700	4.427
12.00	5.803	6.000

100

TIME	VRMS	VINT
0.300	1.450	1.450
0.490	2.100	2.838
0.990	3.000	3.874
1.820	4.000	4.935
12.00	5.742	6.000

260

TIME	VRMS	VINT
0.220	1.450	1.450
0.300	1.750	2.368
0.800	2.300	2.574
1.440	3.800	5.087
12.00	5.780	6.000

420

TIME	VRMS	VINT
0.260	1.450	1.450
0.580	1.850	2.120
1.020	2.500	3.159
1.550	3.850	5.596
12.00	5.766	6.001

580

TIME	VRMS	VINT
0.210	1.450	1.450
0.520	1.850	2.078
0.820	2.150	2.589
1.370	3.500	4.860
12.00	5.770	6.000

750

TIME	VRMS	VINT
0.210	1.450	1.450
0.500	1.800	2.016
0.700	2.600	3.945
1.900	4.200	4.898
12.00	5.753	6.000

910

TIME	VRMS	VINT
0.150	1.450	1.450
0.600	2.100	2.276
0.850	2.800	4.009
1.350	4.300	6.049
12.00	5.834	6.001

1070

TIME	VRMS	VINT
0.140	1.450	1.450
0.400	2.500	2.913
0.740	2.900	3.309
1.120	4.000	5.548
12.00	5.842	6.000

1240

TIME	VRMS	VINT
0.170	1.450	1.450
0.560	2.400	2.651
1.040	3.000	3.825
1.600	4.000	5.385
12.00	5.774	6.001

140

TIME	VRMS	VINT
0.300	1.450	1.450
0.420	1.800	2.467
1.040	3.200	3.871
1.500	3.500	4.098
12.00	5.747	6.000

300

TIME	VRMS	VINT
0.250	1.450	1.450
0.670	2.000	2.265
1.000	3.500	5.385
1.440	4.000	4.952
12.00	5.797	6.000

460

TIME	VRMS	VINT
0.200	1.450	1.450
0.570	2.350	2.715
1.070	2.850	3.330
1.550	3.850	5.455
12.00	5.768	6.001

630

TIME	VRMS	VINT
0.210	1.450	1.450
0.700	2.000	2.194
0.980	3.200	5.083
1.550	4.500	6.120
12.00	5.828	6.000

790

TIME	VRMS	VINT
0.180	1.450	1.450
0.400	1.800	2.042
0.780	2.700	3.399
1.800	4.500	5.492
12.00	5.800	6.000

950

TIME	VRMS	VINT
0.150	1.450	1.450
0.500	2.200	2.452
0.900	3.500	4.636
1.200	4.100	5.522
12.00	5.838	6.000

1110

TIME	VRMS	VINT
0.140	1.450	1.450
0.290	2.300	2.875
0.770	3.100	3.498
1.550	4.100	4.891
12.00	5.790	6.000

1280

TIME	VRMS	VINT
0.190	1.450	1.450
0.290	2.000	2.758
0.940	3.350	3.801
1.800	4.200	5.175
12.00	5.792	6.000

1320

TIME	VRMS	VINT
0.190	1.450	1.450
0.290	2.050	2.882
0.820	3.000	3.410
1.720	3.800	4.404
12.00	5.737	6.000

1360

TIME	VRMS	VINT
0.190	1.450	1.450
0.280	1.950	2.719
0.820	3.100	3.553
1.500	4.200	5.227
12.00	5.806	6.000

1400

TIME	VRMS	VINT
0.190	1.450	1.450
0.380	2.100	2.592
0.710	3.000	3.780
1.370	4.000	4.851
12.00	5.807	6.000

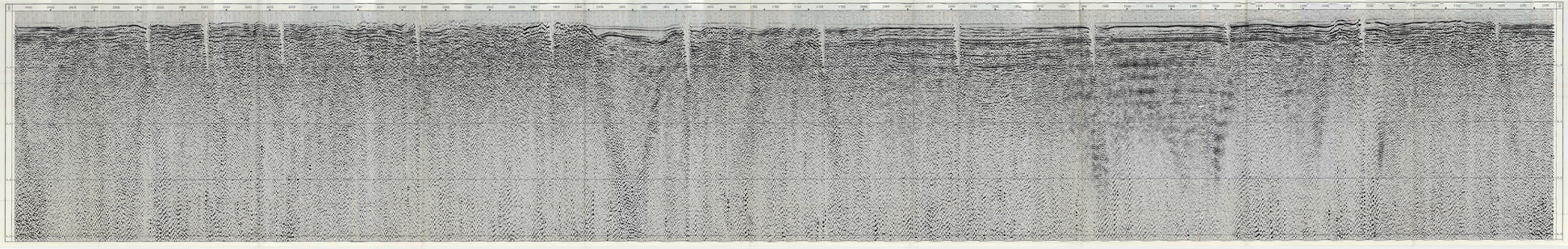
DURHAM UNIVERSITY

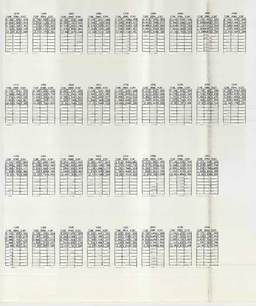
SEISMIC PROCESSING

FARNELLA 1/81 LINE : 6

SYSTEM PARAMETERS

```
***FIELD DATA***
COLLECTED ON               FARNELLA CRUISE NO: 1/81
SOURCE/RECEIVER OFFSET     245 METRES
RECEIVER SPACING           100 METRES
SAMPLING INTERVAL          4 MSEC

***DEMULTIPLEX***

***SORT***
FOLD                       24
COMMON MID-POINT SPACING 50 METRES
RECORDER ERROR REMOVAL

***DBS***
TIME BREAK CORRECTION      64 MSEC
SOURCE DEPTH               10 METRES
RECEIVER DEPTH             10 METRES
POLARITY REVERSAL TRACES 2,5,10 AND 12
BAND-PASS FILTER           5,10 TO 40,60 HZ
VELOCITY FILTER            OPERATOR SIZE 11*11
                           FALSE VELOCITY STRUCTURE
                           TIME (MSEC)      RMS VELOCITY (KM/SEC)
                              400                1.4
                             1000                2.1
                             2000                3.0
MUTE                       TRACE NUMBER     TIME (MSEC)
                                4                500
                                7               1100
                               24               2800

***STACK***
RESAMPLED PRIOR TO NMO     1 MSEC
NMO AND STACK              24 FOLD

***DAS***
BAND-PASS FILTER           5,10 TO 40,60 HZ
GEOMETRIC SPEADING CORRECTION
DECONVOLUTION              PREDICTION ERROR
                           AUTOCORR. DESIGN       2000 MSEC FROM 0 MSEC
                           ACTIVE LENGTH           400 MSEC
                           PREDICTION LAG          100 MSEC
                           PRE-WHITENING             1 %
TRACE NORMALIZATION        UNIT ENERGY

***DISPLAY***
GAIN                       150
CLIP                       -0.1 TO 0.1 INCHES
TRACE SPACING              0.05 INCHES
VERTICAL SCALE             2.5 INCHES/SEC

PROCESSED BY               R.W.HOBBS
```

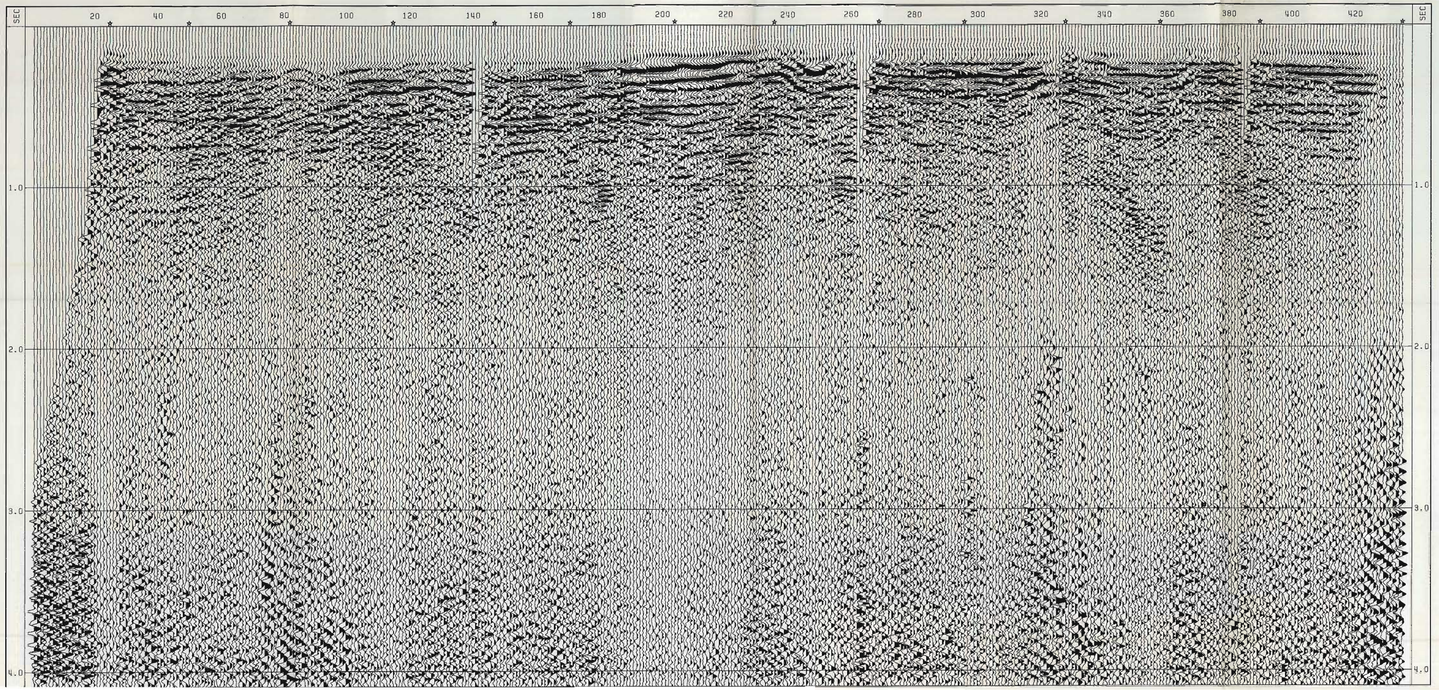


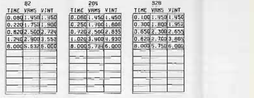


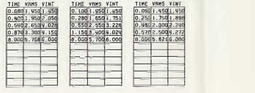
DURHAM UNIVERSITY
SEISMIC PROCESSING
FARNELLA 1/81
LINE : B

SYSTEM PARAMETERS

FARNELLA 1/81 LINE : 4 START GATHER NO. 1 END GATHER NO. 435
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.050 INCHES TRACE LENGTH 10.24 INCHES
PLOTTING GAIN 1.500E+02 LOW CLIP LEVEL -0.100 INCHES HIGH CLIP LEVEL 0.100 INCHES







DURHAM UNIVERSITY

SEISMIC PROCESSING

FARNELLA 1/81 LINE : 4

SYSTEM PARAMETERS

FIELD DATA

COLLECTED ON	FARNELLA CRUISE NO: 1/81
SOURCE/RECEIVER OFFSET	245 METRES
RECEIVER SPACING	100 METRES
SAMPLING INTERVAL	4 MSEC

DEMULTIPLEX

SORT

FOLD	24
COMMON MID-POINT SPACING	50 METRES
RECORDER ERROR REMOVAL	

DBS

TIME BREAK CORRECTION	64 MSEC
SOURCE DEPTH	10 METRES
RECEIVER DEPTH	10 METRES
POLARITY REVERSAL TRACES	2,5,10 AND 12
BAND-PASS FILTER	5,10 TO 40,60 HZ
VELOCITY FILTER	OPERATOR SIZE 11*11

FALSE VELOCITY STRUCTURE

TIME (MSEC)	RMS VELOCITY (KM/SEC)
400	1.4
1000	2.1
2000	3.0

MUTE	TRACE NUMBER	TIME (MSEC)
	4	500
	7	1100
	24	2800

STACK

RESAMPLED PRIOR TO NMO	1 MSEC
NMO AND STACK	24 FOLD

DAS

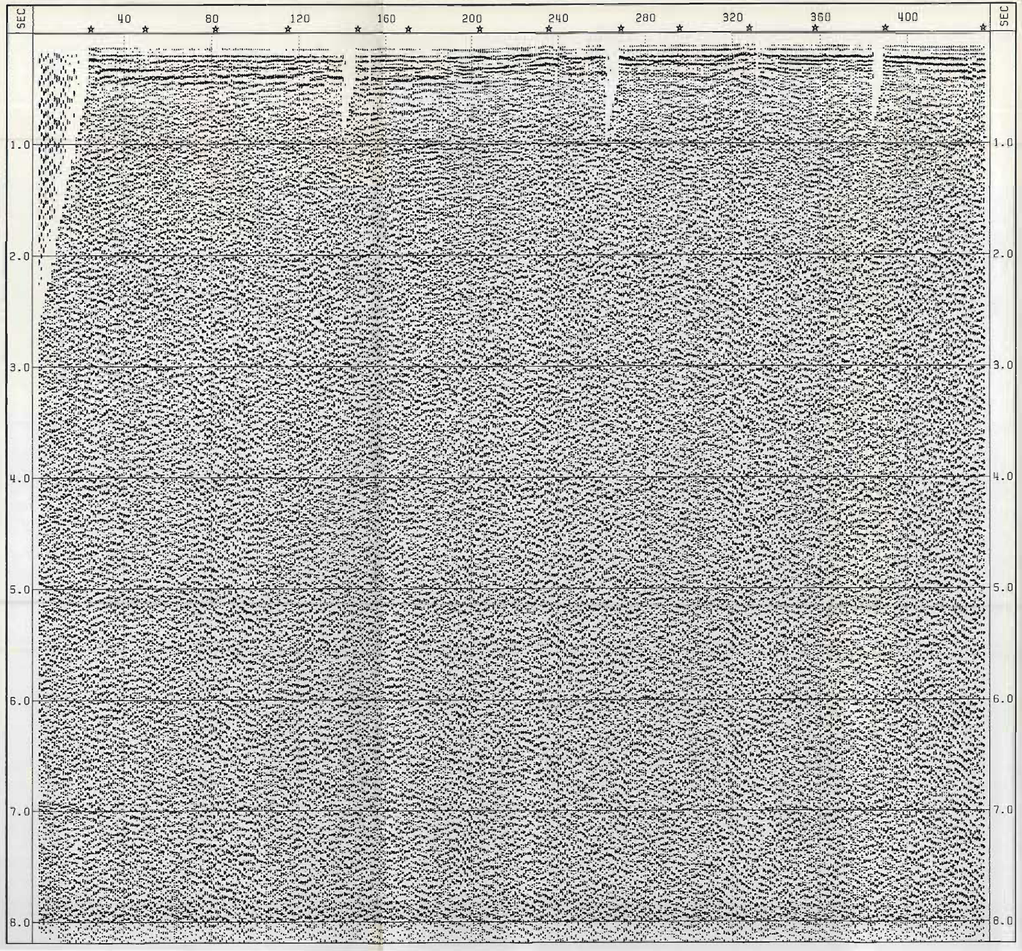
BAND-PASS FILTER	5,10 TO 40.60 HZ
GEOMETRIC SPREADING CORRECTION	
DECONVOLUTION	PREDICTION ERROR
	AUTOCORR. DESIGN 2000 MSEC FROM 0 MSEC
	ACTIVE LENGTH 400 MSEC
	PREDICTION LAG 100 MSEC
	PRE-WHITENING 1 %
TRACE NORMALIZATION	UNIT ENERGY

DISPLAY

GAIN	150
CLIP	-0.1 TO 0.1 INCHES
TRACE SPACING	0.05 INCHES
VERTICAL SCALE	2,5 INCHES/SEC

PROCESSED BY R.W.HOBBS

FARNELLA 1/81 LINE : 4 START GATHER NO. 1 END GATHER NO. 435
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.025 INCHES TRACE LENGTH 10.24 INCHES 50 SAMPLE AGC APPLIED
PLOTTING GAIN 2.500E+00 LOW CLIP LEVEL 0.005 INCHES HIGH CLIP LEVEL 0.050 INCHES



25

TIME	VRMS	VINT
0.100	1.450	1.450
0.370	1.800	1.818
0.530	2.400	3.409
0.750	3.600	5.505
6.000	5.817	6.000

147

TIME	VRMS	VINT
0.080	1.450	1.450
0.220	1.700	1.826
0.700	2.500	2.791
0.850	3.150	5.202
8.000	5.706	5.999

269

TIME	VRMS	VINT
0.100	1.450	1.450
0.200	1.650	1.826
0.940	2.250	2.595
0.950	3.300	4.104
6.000	5.746	6.000

390

TIME	VRMS	VINT
0.100	1.450	1.450
0.220	1.700	1.683
0.480	2.130	2.387
0.650	2.700	3.927
6.000	5.832	6.000

50

TIME	VRMS	VINT
0.080	1.450	1.450
0.200	1.700	1.696
0.350	2.150	3.186
0.770	3.400	4.024
8.000	5.801	5.999

171

TIME	VRMS	VINT
0.080	1.450	1.450
0.230	1.700	1.819
0.600	2.300	2.502
1.150	3.500	5.308
8.000	5.708	6.000

296

TIME	VRMS	VINT
0.100	1.450	1.450
0.220	1.700	1.863
0.580	2.300	2.598
0.860	2.950	3.871
8.000	5.750	6.000

435

TIME	VRMS	VINT
0.100	1.450	1.450
0.200	1.700	1.818
0.450	2.400	2.636
0.750	3.350	4.900
6.000	5.803	6.000

82

TIME	VRMS	VINT
0.080	1.450	1.450
0.220	1.750	1.800
0.820	2.500	3.724
1.240	2.900	3.553
6.000	5.838	6.000

204

TIME	VRMS	VINT
0.080	1.450	1.450
0.250	1.700	1.806
0.720	2.500	2.835
1.020	3.400	4.930
8.000	5.730	6.000

328

TIME	VRMS	VINT
0.100	1.450	1.450
0.300	1.800	1.952
0.630	2.300	2.655
0.820	2.700	3.885
8.000	5.750	6.000

116

TIME	VRMS	VINT
0.080	1.450	1.450
0.900	1.950	2.250
0.940	2.650	4.028
0.870	3.300	4.150
8.000	5.766	6.000

236

TIME	VRMS	VINT
0.100	1.450	1.450
0.260	1.650	1.751
0.950	2.550	3.228
1.150	3.900	9.024
8.000	5.700	6.000

358

TIME	VRMS	VINT
0.090	1.450	1.450
0.250	1.750	1.898
0.980	2.600	2.240
0.970	2.330	4.272
8.000	5.826	6.000

DURHAM UNIVERSITY

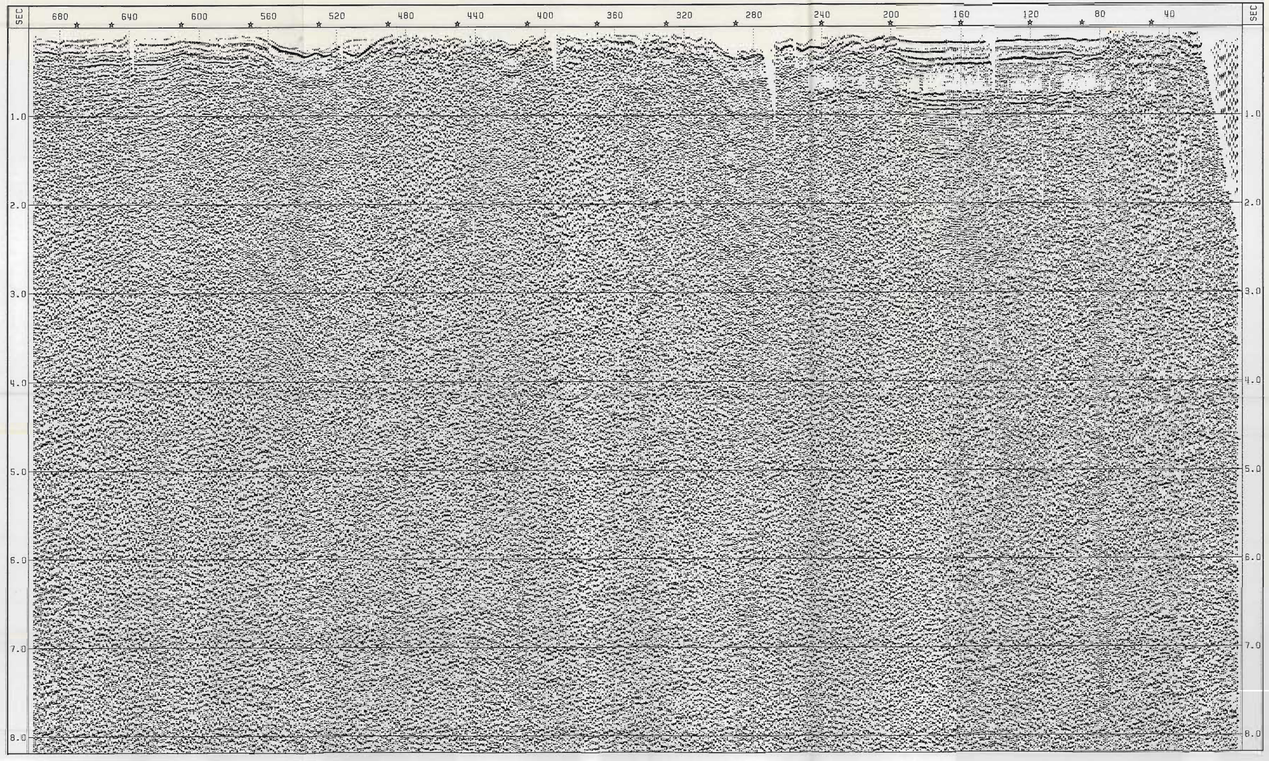
SEISMIC PROCESSING

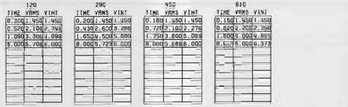
FARNELLA 1/81 LINE : 4

SYSTEM PARAMETERS

FIELD DATA		
COLLECTED ON	FARNELLA CRUISE NO: 1/81	
SOURCE/RECEIVER OFFSET	245 METRES	
RECEIVER SPACING	100 METRES	
SAMPLING INTERVAL	4 MSEC	
DEMULTIPLEX		
SORT		
FOLD	24	
COMMON MID-POINT SPACING	50 METRES	
RECORDER ERROR REMOVAL		
OBS		
TIME BREAK CORRECTION	64 MSEC	
SOURCE DEPTH	10 METRES	
RECEIVER DEPTH	10 METRES	
POLARITY REVERSAL TRACES	2,5,10 AND 12	
BAND-PASS FILTER	5,10 TO 40,60 HZ	
TIME RAMP	EXP(T)	
DECONVOLUTION	PREDICTION ERROR	
	AUTOCORR. DESIGN	4000 MSEC FROM 0 MSEC
	ACTIVE LENGTH	400 MSEC
	PREDICTION LAG	40 MSEC
	PRE-WHITENING	1 %
REMOVE TIME RAMP		
MUTE	TRACE NUMBER TIME (MSEC)	
	2 500	
	24 2700	
STACK		
RESAMPLED PRIOR TO NMO	2 MSEC	
NMO AND STACK	24 FOLD	
DAS		
BAND-PASS FILTER	5,10 TO 40,60 HZ	
GEOMETRIC SPREADING CORRECTION		
DECONVOLUTION	PREDICTION ERROR	
	AUTOCORR. DESIGN	2000 MSEC FROM 0 MSEC
	ACTIVE LENGTH	400 MSEC
	PREDICTION LAG	100 MSEC
	PRE-WHITENING	1 %
TRACE NORMALIZATION	UNIT ENERGY	
DISPLAY		
A.G.C. APPLIED	200 MSEC WINDOW	
GAIN	2.5	
CLIP	0.005 TO 0.05 INCHES	
TRACE SPACING	0.025 INCHES	
VERTICAL SCALE	1.25 INCHES/SEC	
PROCESSED BY	R.W.HOBBS	

FARNELLA 1/81 LINE : 10 START GATHER NO. 1 END GATHER NO. 695
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.025 INCHES TRACE LENGTH 10.24 INCHES 50 SAMPLE AGC APPLIED
PLOTTING GAIN 2.500E+00 LOW CLIP LEVEL 0.005 INCHES HIGH CLIP LEVEL 0.050 INCHES





DURHAM UNIVERSITY

SEISMIC PROCESSING

FARNELLA 1/81 LINE : 10

SYSTEM PARAMETERS

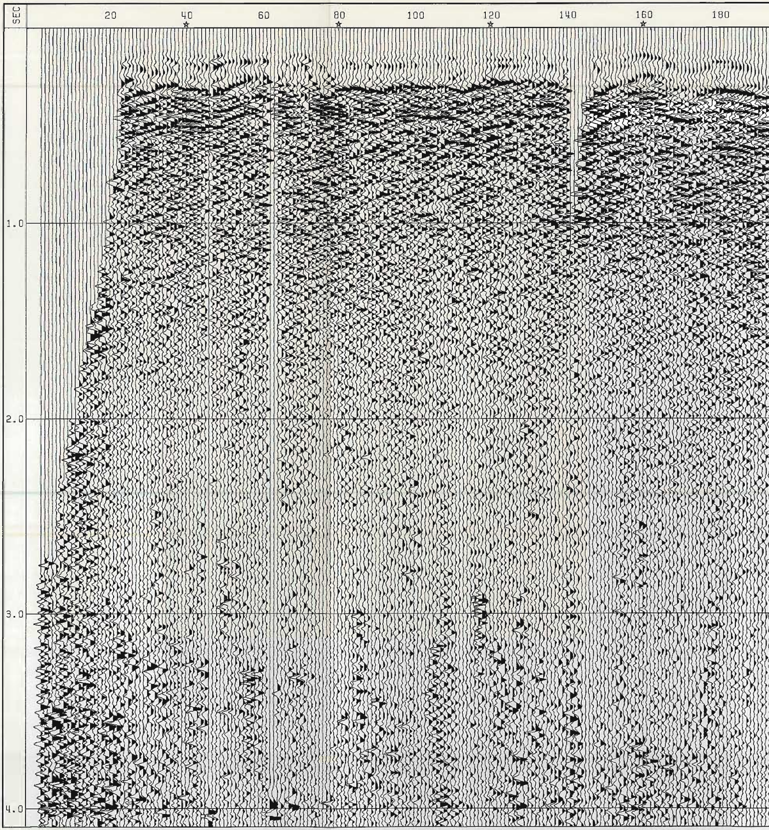
FIELD DATA		
COLLECTED ON	FARNELLA CRUISE NO: 1/81	
SOURCE/RECEIVER OFFSET	245 METRES	
RECEIVER SPACING	100 METRES	
SAMPLING INTERVAL	4 MSEC	
DEMULTIPLEX		
SORT		
FOLD	24	
COMMON MID-POINT SPACING	50 METRES	
RECORDER ERROR REMOVAL		
OBS		
TIME BREAK CORRECTION	64 MSEC	
SOURCE DEPTH	10 METRES	
RECEIVER DEPTH	10 METRES	
POLARITY REVERSAL TRACES	1,2,5,10,12 AND 20	
BAND-PASS FILTER	5,10 TO 40,60 HZ	
TIME RAMP	EXP(T)	
DECONVOLUTION	PREDICTION ERROR	
	AUTOCORR. DESIGN	4000 MSEC FROM 0 MSEC
	ACTIVE LENGTH	400 MSEC
	PREDICTION LAG	40 MSEC
	PRE-WHITENING	1 %
REMOVE TIME RAMP		
MUTE	TRACE NUMBER TIME (MSEC)	
	2 500	
	24 2700	
STACK		
RESAMPLED PRIOR TO NMO	2 MSEC	
NMO AND STACK	24 FOLD	
OBS		
BAND-PASS FILTER	5,10 TO 40,60 HZ	
GEOMETRIC SPREDING CORRECTION		
DECONVOLUTION	PREDICTION ERROR	
	AUTOCORR. DESIGN	2000 MSEC FROM 0 MSEC
	ACTIVE LENGTH	400 MSEC
	PREDICTION LAG	100 MSEC
	PRE-WHITENING	1 %
TRACE NORMALIZATION	UNIT ENERGY	
DISPLAY		
A.G.C. APPLIED	200 MSEC WINDOW	
GAIN	2.5	
CLIP	0.005 TO 0.05 INCHES	
TRACE SPACING	0.025 INCHES	
VERTICAL SCALE	1.25 INCHES/SEC	
PROCESSED BY	R.W.HOBBS	

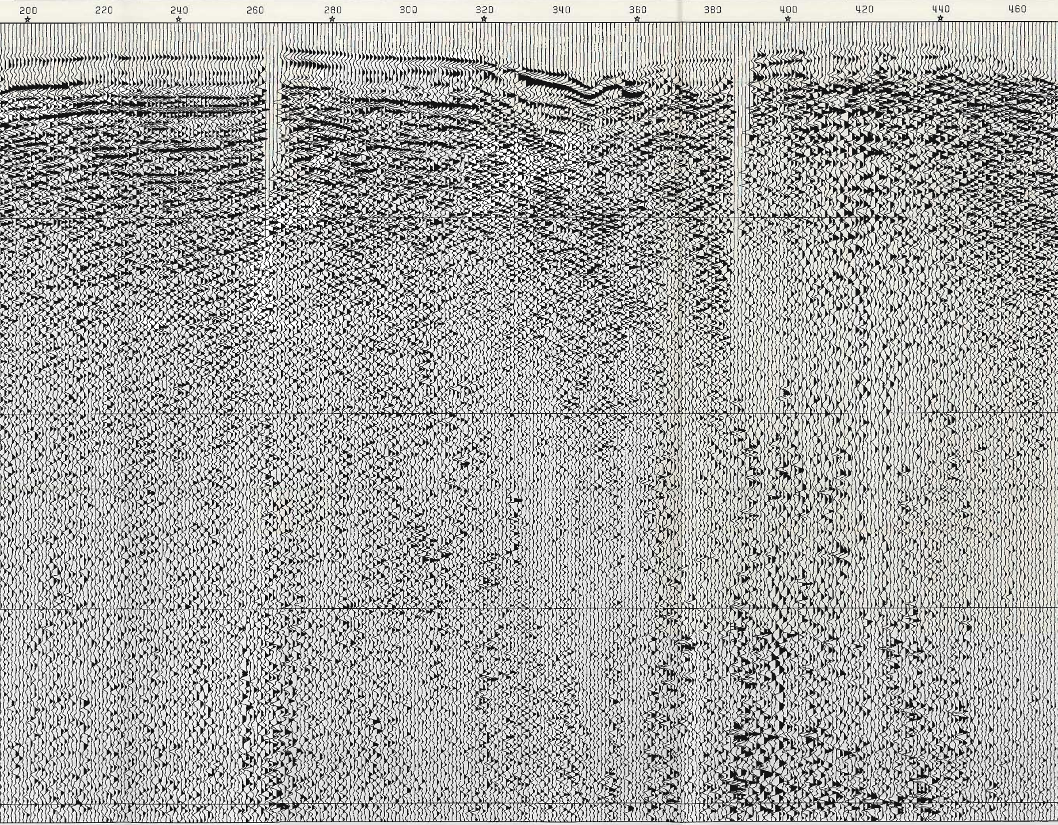
FARNELLA 1/81 LINE : 9 START GATHER NO. 1 END GATHER NO. 1014

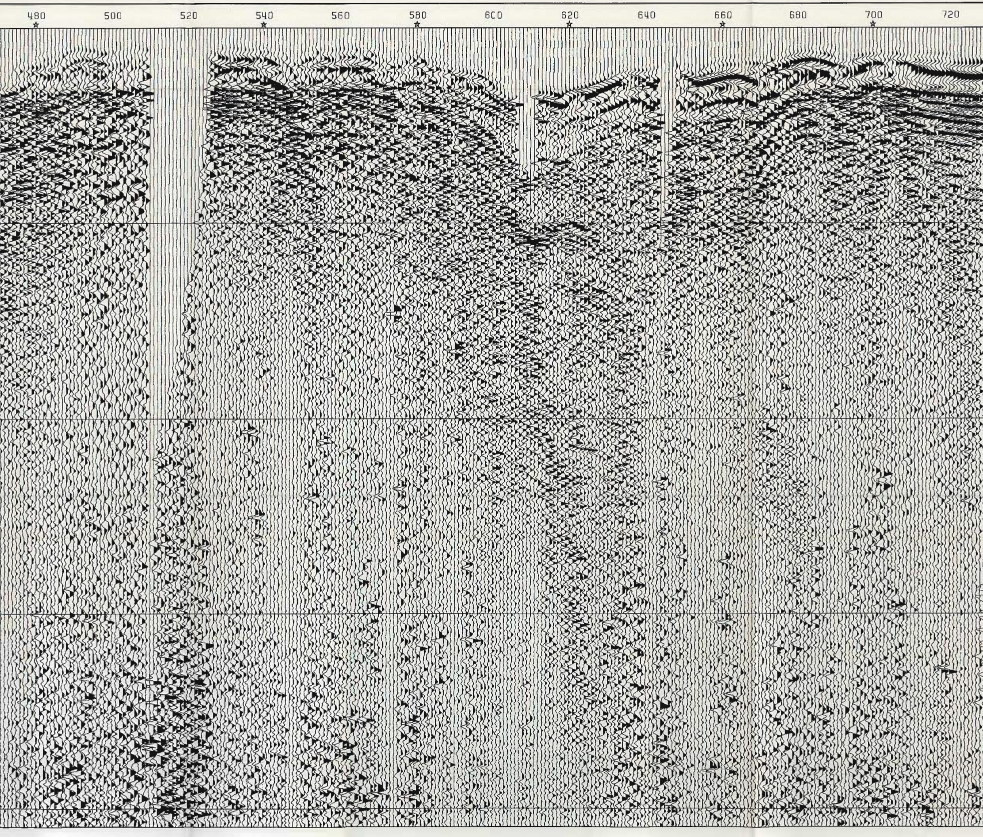
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED

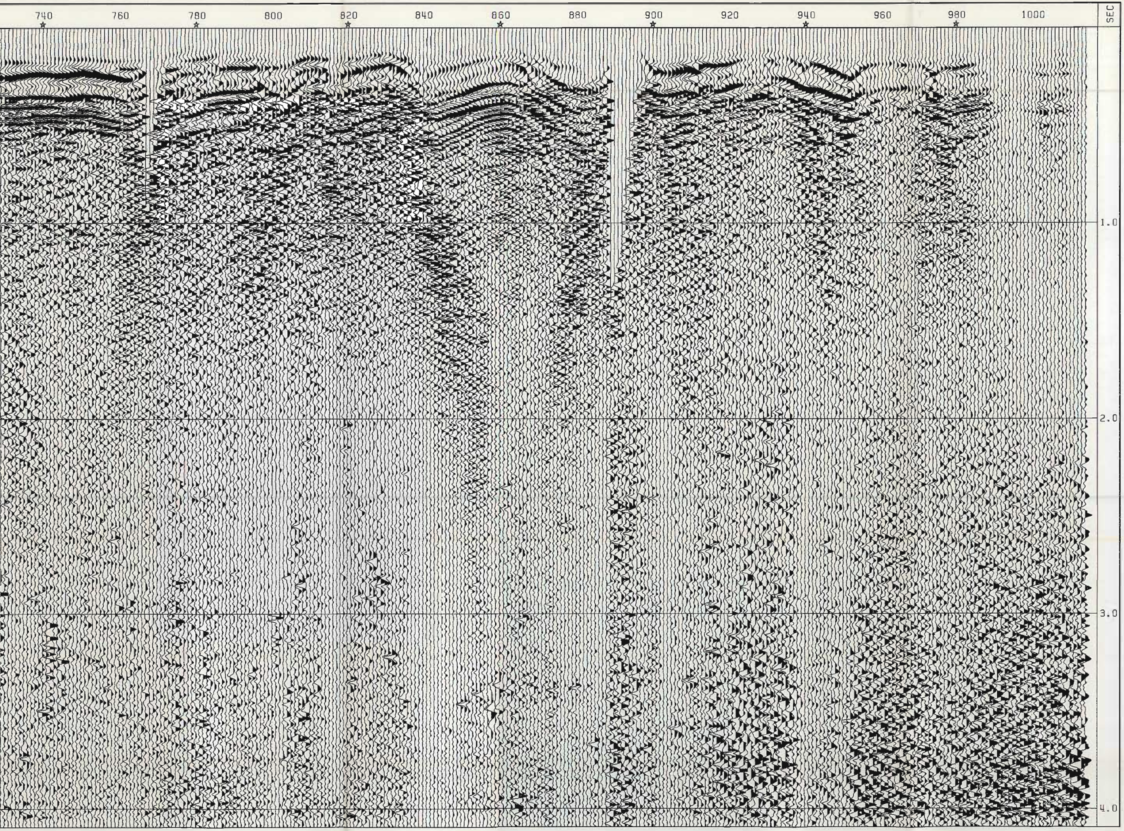
TRACE SPACING 0.050 INCHES TRACE LENGTH 10.24 INCHES

PLOTTING GAIN 1.500E+02 LOW CLIP LEVEL -0.100 INCHES HIGH CLIP LEVEL 0.100 INCHES









40

TIME	VRMS	VINT
0.200	1.450	1.450
0.420	2.500	3.166
0.600	3.700	5.572
2.640	5.800	6.046
12.00	5.914	6.000

80

TIME	VRMS	VINT
0.200	1.450	1.450
0.290	2.400	3.727
0.650	3.900	4.777
2.350	5.300	5.746
12.00	5.869	5.999

120

TIME	VRMS	VINT
0.180	1.450	1.450
0.290	2.300	3.241
0.550	3.100	3.796
2.050	5.200	5.782
12.00	5.871	8.000

160

TIME	VRMS	VINT
0.180	1.450	1.450
0.320	2.300	3.064
0.600	3.500	4.495
1.950	5.200	5.798
12.00	5.877	6.000

200

TIME	VRMS	VINT
0.200	1.450	1.450
0.390	2.300	2.940
0.970	3.800	4.538
2.550	5.300	6.039
12.00	5.858	6.000

240

TIME	VRMS	VINT
0.200	1.450	1.450
0.350	2.400	3.261
0.680	3.100	3.700
1.950	5.300	6.183
12.00	5.892	6.000

280

TIME	VRMS	VINT
0.200	1.450	1.450
0.400	2.300	2.912
0.840	4.500	5.818
2.150	5.500	6.055
12.00	5.914	6.001

320

TIME	VRMS	VINT
0.250	1.450	1.450
0.430	2.100	2.760
0.610	3.200	4.916
1.400	4.300	4.986
12.00	5.825	5.997

360

TIME	VRMS	VINT
0.310	1.450	1.450
0.430	2.500	4.119
0.900	4.000	4.992
1.950	5.300	6.201
12.00	5.892	6.000

400

TIME	VRMS	VINT
0.200	1.450	1.450
0.380	3.500	4.850
0.890	4.500	5.120
1.640	5.000	5.535
12.00	5.873	6.000

440

TIME	VRMS	VINT
0.180	1.450	1.450
0.290	2.800	4.151
0.800	4.600	5.360
2.000	5.483	8.000
12.00	5.917	6.000

480

TIME	VRMS	VINT
0.250	1.450	1.450
0.350	2.500	4.077
0.650	3.500	4.387
1.900	5.300	6.027
12.00	5.895	6.000

540

TIME	VRMS	VINT
0.210	1.450	1.450
0.300	2.300	3.568
0.650	3.500	4.266
1.770	5.300	6.106
12.00	5.902	6.000

580

TIME	VRMS	VINT
0.220	1.450	1.450
0.270	2.100	3.816
0.650	3.500	4.221
2.100	5.400	6.061
12.00	5.899	6.000

620

TIME	VRMS	VINT
0.350	1.450	1.450
0.540	2.900	4.475
0.900	3.800	4.446
1.930	5.000	5.960
12.00	5.851	6.000

660

TIME	VRMS	VINT
0.300	1.450	1.450
0.450	2.300	3.415
0.850	3.900	5.135
1.850	5.000	5.772
12.00	5.857	6.000

700

TIME	VRMS	VINT
0.200	1.450	1.450
0.290	2.200	3.305
0.570	3.100	3.814
1.980	5.000	5.588
12.00	5.847	6.000

740

TIME	VRMS	VINT
0.200	1.450	1.450
0.400	2.300	2.912
0.580	3.500	4.711
1.800	5.200	6.002
12.00	5.887	6.000

780

TIME	VRMS	VINT
0.250	1.450	1.450
0.410	2.200	3.019
0.620	3.000	4.138
1.950	5.200	5.954
12.00	5.877	6.000

820

TIME	VRMS	VINT
0.200	1.450	1.450
0.400	2.300	2.912
0.680	3.700	5.069
1.250	4.900	6.027
12.00	5.895	6.000

860

TIME	VRMS	VINT
0.200	1.500	1.500
0.300	2.100	2.955
0.540	3.500	4.696
1.500	5.100	5.809
12.00	5.895	6.000

900

TIME	VRMS	VINT
0.200	1.500	1.500
0.300	2.100	2.955
0.700	3.500	4.258
1.400	4.900	5.981
12.00	5.882	8.000

940

TIME	VRMS	VINT
0.200	1.500	1.500
0.520	2.600	3.095
0.870	3.200	3.926
1.900	5.000	5.121
12.00	5.853	6.000

980

TIME	VRMS	VINT
0.260	1.500	1.500
0.350	2.500	4.220
0.870	3.700	4.324
1.870	5.000	5.903
12.00	5.855	6.000

DURHAM UNIVERSITY

SEISMIC PROCESSING

FARNELLA 1/81 LINE : 9

SYSTEM PARAMETERS

FIELD DATA

COLLECTED ON	FARNELLA CRUISE NO: 1/81
SOURCE/RECEIVER OFFSET	245 METRES
RECEIVER SPACING	100 METRES
SAMPLING INTERVAL	4 MSEC

DEMULTIPLEX

SORT

FOLD	24
COMMON MID-POINT SPACING	50 METRES
RECORDER ERROR REMOVAL	

DBS

TIME BREAK CORRECTION	64 MSEC
SOURCE DEPTH	10 METRES
RECEIVER DEPTH	10 METRES
POLARITY REVERSAL TRACES	1,2,5,10,12 AND 20
BAND-PASS FILTER	5,10 TO 40,60 HZ
VELOCITY FILTER	OPERATOR SIZE 11×11
	FALSE VELOCITY STRUCTURE

TIME (MSEC)	RMS VELOCITY (KM/SEC)
400	1.4
1000	2.1
2000	3.0

MUTE

TRACE NUMBER	TIME (MSEC)
4	500
7	1100
24	2800

STACK

RESAMPLED PRIOR TO NMO	1 MSEC
NMO AND STACK	24 FOLD

DAS

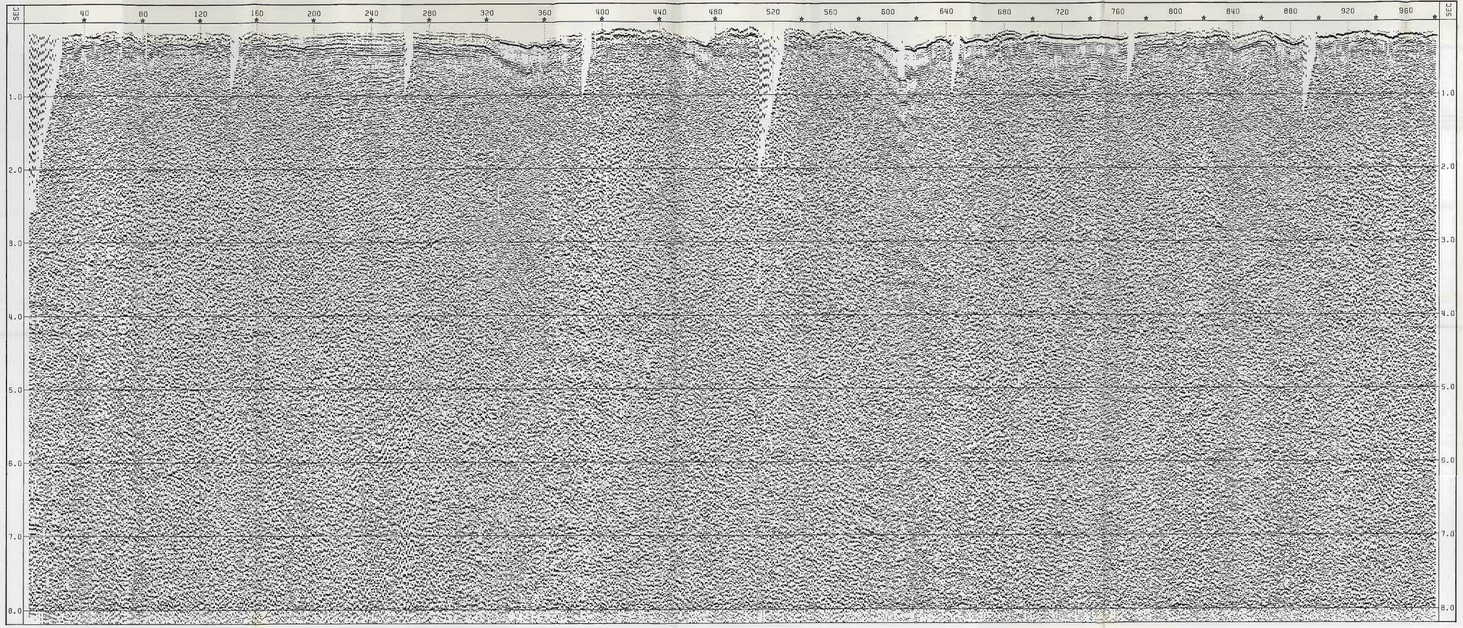
BAND-PASS FILTER	5,10 TO 40,60 HZ	
GEOMETRIC SPREADING CORRECTION		
DECONVOLUTION	PREDICTION ERROR	
	AUTOCORR. DESIGN	2000 MSEC FROM 0 MSEC
	ACTIVE LENGTH	400 MSEC
	PREDICTION LAG	100 MSEC
	PRE-WHITENING	1 %
TRACE NORMALIZATION	UNIT ENERGY	

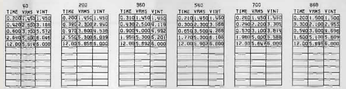
DISPLAY

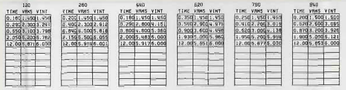
GAIN	150
CLIP	-0.1 TO 0.1 INCHES
TRACE SPACING	0.05 INCHES
VERTICAL SCALE	2.5 INCHES/SEC

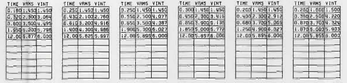
PROCESSED BY R.W.HOBBS

FARNELLA 1/81 LINE : 9 START GATHER NO. 1 END GATHER NO. 980
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.025 INCHES TRACE LENGTH 10.24 INCHES 50 SAMPLE AGC APPLIED
PLOTTING GAIN 2.500E+00 LOW CLIP LEVEL 0.005 INCHES HIGH CLIP LEVEL 0.050 INCHES







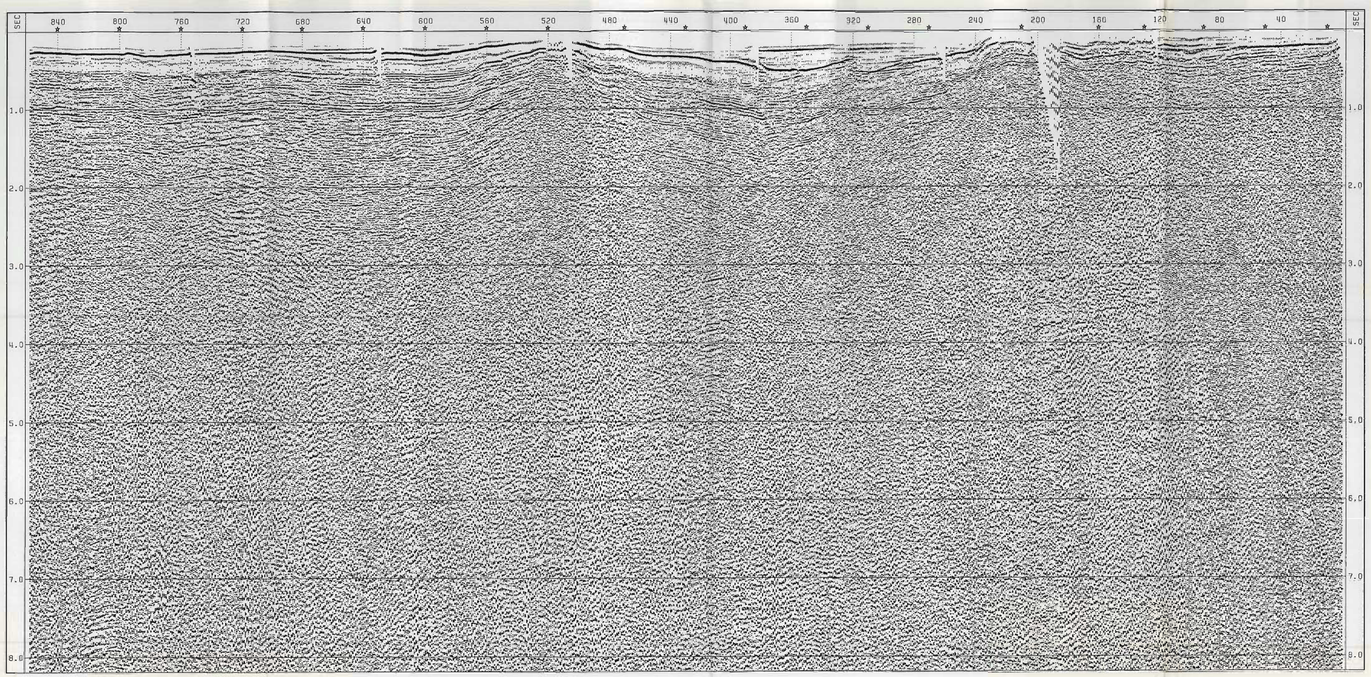


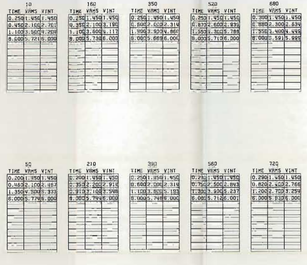
DURHAM UNIVERSITY
SEISMIC PROCESSING
FARNELLA 1/81 LINE : 9

SYSTEM PARAMETERS

FIELD DATA		
COLLECTED ON	FARNELLA CRUISE NO: 1/81	
SOURCE/RECEIVER OFFSET	245 METRES	
RECEIVER SPACING	100 METRES	
SAMPLING INTERVAL	4 MSEC	
DEMULTIPLEX		
SORT		
FOLD	24	
COMMON MID-POINT SPACING	50 METRES	
RECORDER ERROR REMOVAL		
DBS		
TIME BREAK CORRECTION	64 MSEC	
SOURCE DEPTH	10 METRES	
RECEIVER DEPTH	10 METRES	
POLARITY REVERSAL TRACES	1,2,5,10,12 AND 20	
BAND-PASS FILTER	5,10 TO 40,60 HZ	
TIME RAMP	EXP(T)	
DECONVOLUTION	PREDICTION ERROR	
	AUTOCORR. DESIGN	4000 MSEC FROM 0 MSEC
	ACTIVE LENGTH	400 MSEC
	PREDICTION LAG	40 MSEC
	PRE-WHITENING	1 %
REMOVE TIME RAMP		
MUTE	TRACE NUMBER TIME (MSEC)	
	2 500	
	24 2700	
STACK		
RESAMPLED PRIOR TO NMO	2 MSEC	
NMO AND STACK	24 FOLD	
DBS		
BAND-PASS FILTER	5,10 TO 40,60 HZ	
GEOMETRIC SPREADING CORRECTION		
DECONVOLUTION	PREDICTION ERROR	
	AUTOCORR. DESIGN	2000 MSEC FROM 0 MSEC
	ACTIVE LENGTH	400 MSEC
	PREDICTION LAG	100 MSEC
	PRE-WHITENING	1 %
TRACE NORMALIZATION	UNIT ENERGY	
DISPLAY		
A.G.C. APPLIED	200 MSEC WINDOW	
GAIN	2.5	
CLIP	0.005 TO 0.05 INCHES	
TRACE SPACING	0.025 INCHES	
VERTICAL SCALE	1.25 INCHES/SEC	
PROCESSED BY	A.W.HOBBS	

FARNELLA 1/81 LINE : 8 START GATHER NO. 1 END GATHER NO. 858
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.025 INCHES TRACE LENGTH 10.24 INCHES 50 SAMPLE AGC APPLIED
PLOTTING GAIN 2.500E+00 LOW CLIP LEVEL 0.005 INCHES HIGH CLIP LEVEL 0.050 INCHES





DURHAM UNIVERSITY
SEISMIC PROCESSING
FARNELLA 1/81 LINE : 8

SYSTEM PARAMETERS

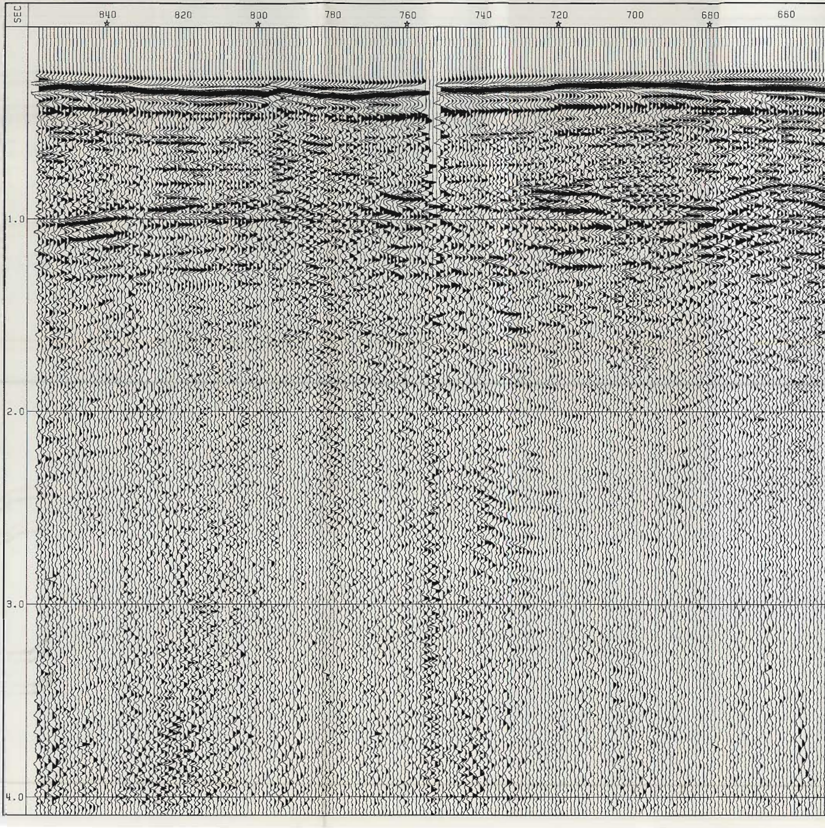
FIELD DATA	
COLLECTED ON	FARNELLA CRUISE NO: 1/81
SOURCE/RECEIVER OFFSET	245 METRES
RECEIVER SPACING	100 METRES
SAMPLING INTERVAL	4 MSEC
DEMULTIPLEX	
SORT	
FOLD	24
COMMON MID-POINT SPACING	50 METRES
RECORDER ERROR REMOVAL	
OBS	
TIME BREAK CORRECTION	64 MSEC
SOURCE DEPTH	10 METRES
RECEIVER DEPTH	10 METRES
POLARITY REVERSAL TRACES	2,6,10 AND 12
BAND-PASS FILTER	5.10 TO 40.60 HZ
TIME RAMP	EXP (T)
DECONVOLUTION	PREDICTION ERROR
	AUTOCORR. DESIGN 4000 MSEC FROM 0 MSEC
	ACTIVE LENGTH 400 MSEC
	PREDICTION LAG 40 MSEC
	PRE-WHITENING 1 %
REMOVE TIME RAMP	
MUTE	TRACE NUMBER TIME (MSEC)
	2 500
	24 2700
STACK	
RESAMPLED PRIOR TO NMO	2 MSEC
NMO AND STACK	24 FOLD
DAS	
BAND-PASS FILTER	5.10 TO 40.60 HZ
GEOMETRIC SPREADING CORRECTION	
DECONVOLUTION	PREDICTION ERROR
	AUTOCORR. DESIGN 2000 MSEC FROM 0 MSEC
	ACTIVE LENGTH 400 MSEC
	PREDICTION LAG 100 MSEC
	PRE-WHITENING 1 %
TRACE NORMALIZATION	UNIT ENERGY
DISPLAY	
A.G.C. APPLIED	200 MSEC WINDOW
GAIN	2.5
CLIP	0.005 TO 0.05 INCHES
TRACE SPACING	0.025 INCHES
VERTICAL SCALE	1.25 INCHES/SEC
PROCESSED BY	R.W.HOBBS

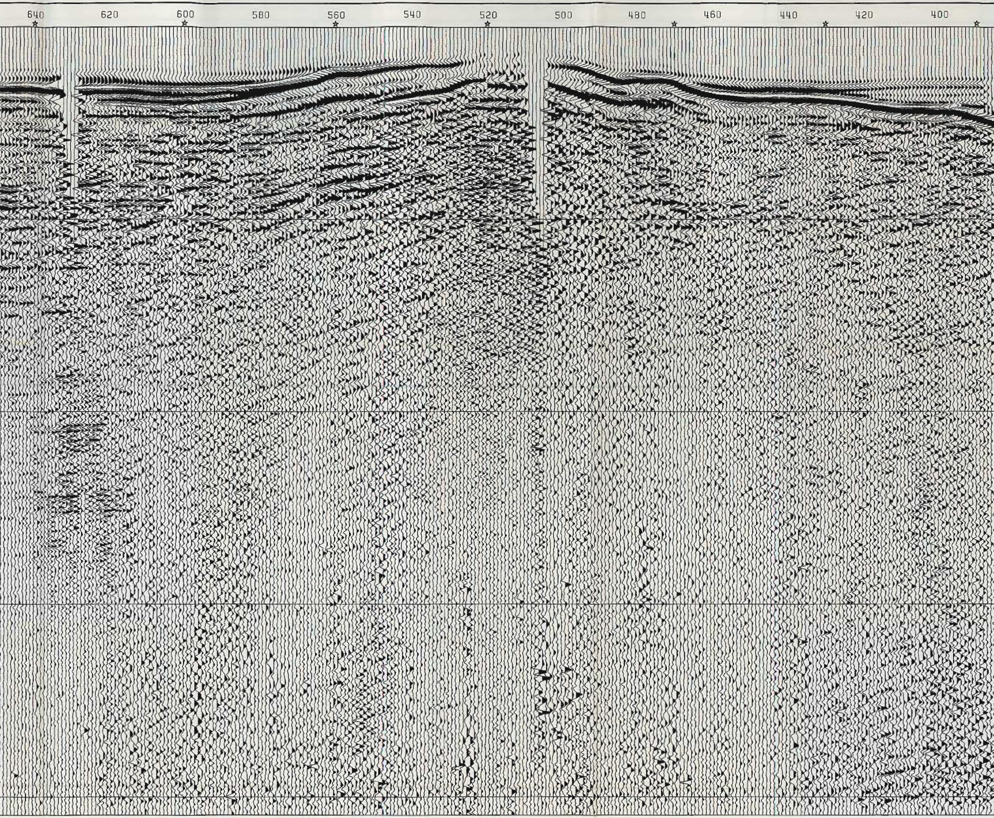
FARNELLA 1/81 LINE : 8 START GATHER NO. 1 END GATHER NO. 858

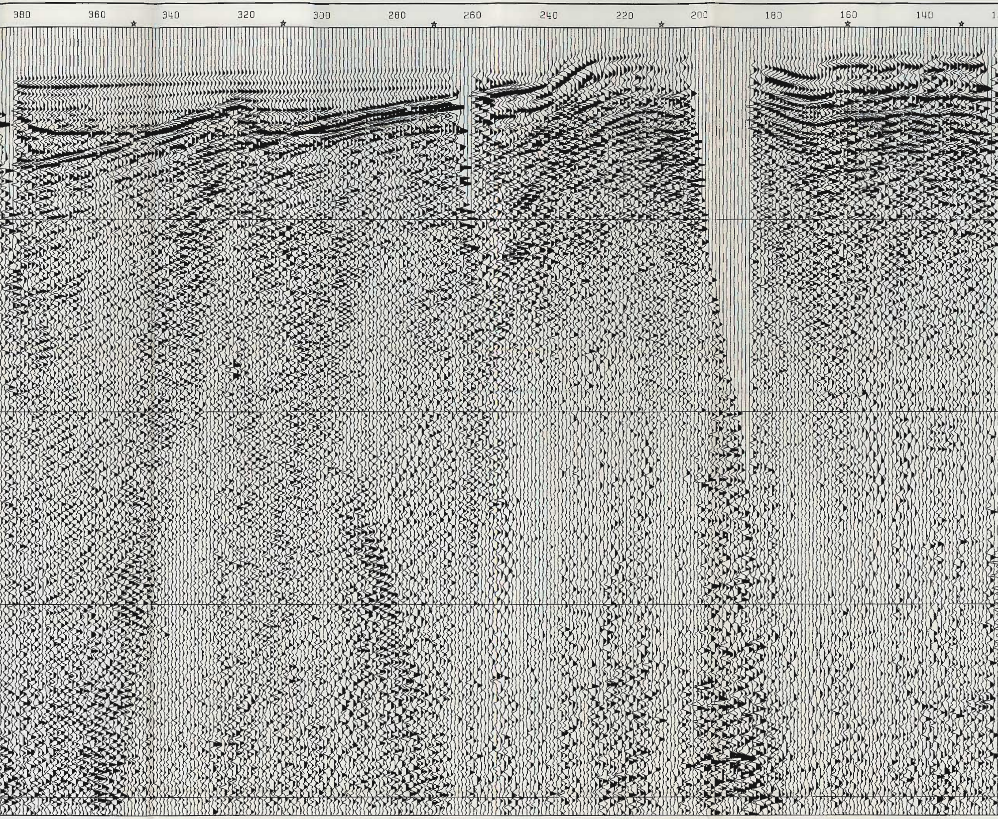
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED

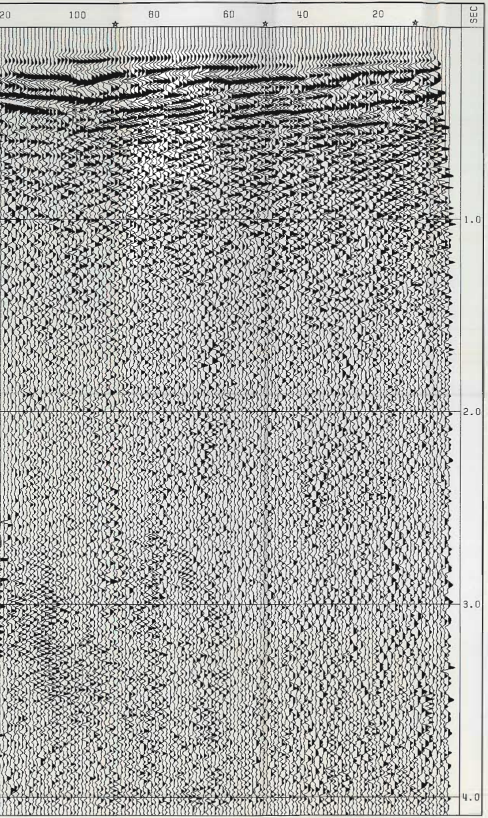
TRACE SPACING 0.050 INCHES TRACE LENGTH 10.24 INCHES

PLOTTING GAIN 1.500E+02 LOW CLIP LEVEL -0.100 INCHES HIGH CLIP LEVEL 0.100 INCHES









10

TIME	VRMS	VINT
0.250	1.450	1.450
0.450	2.100	2.701
1.100	3.500	4.204
8.000	5.721	6.000

160

TIME	VRMS	VINT
0.250	1.450	1.450
0.350	2.100	3.190
1.100	3.600	4.117
8.000	5.730	6.000

350

TIME	VRMS	VINT
0.250	1.450	1.450
0.600	2.000	2.314
1.400	3.900	4.860
8.000	5.689	6.000

520

TIME	VRMS	VINT
0.250	1.450	1.450
0.870	2.600	2.939
1.550	4.300	5.788
8.000	5.710	6.000

680

TIME	VRMS	VINT
0.300	1.450	1.450
0.880	2.300	2.634
1.550	3.400	4.449
8.000	5.591	5.999

840

TIME	VRMS	VINT
0.300	1.450	1.450
0.950	2.600	2.985
2.100	4.000	4.861
8.000	5.545	6.000

50

TIME	VRMS	VINT
0.200	1.450	1.450
0.460	2.100	2.487
1.350	4.500	5.333
8.000	5.774	6.000

210

TIME	VRMS	VINT
0.200	1.450	1.450
0.350	2.200	2.914
0.910	3.100	3.546
8.000	5.744	6.000

390

TIME	VRMS	VINT
0.250	1.450	1.450
0.600	2.000	2.314
1.100	3.800	5.193
8.000	5.748	6.000

560

TIME	VRMS	VINT
0.230	1.450	1.450
0.750	2.500	2.843
1.300	3.900	5.237
8.000	5.712	6.001

720

TIME	VRMS	VINT
0.290	1.450	1.450
0.820	2.400	2.786
1.200	2.700	3.254
8.000	5.630	6.000

90

TIME	VRMS	VINT
0.250	1.450	1.450
0.490	2.600	3.408
1.050	4.000	4.908
8.000	5.777	6.000

270

TIME	VRMS	VINT
0.200	1.450	1.450
0.520	2.300	2.699
0.870	2.900	3.512
8.000	5.745	6.001

430

TIME	VRMS	VINT
0.300	1.450	1.450
0.880	2.400	2.786
1.100	3.000	4.686
8.000	5.682	6.000

600

TIME	VRMS	VINT
0.210	1.450	1.450
0.400	2.500	3.292
0.950	3.500	4.076
8.000	5.760	6.000

760

TIME	VRMS	VINT
0.300	1.450	1.450
0.900	2.200	2.492
1.850	3.500	4.598
8.000	5.577	6.000

130

TIME	VRMS	VINT
0.250	1.450	1.450
0.680	2.900	3.475
1.130	3.700	4.655
8.000	5.731	6.000

310

TIME	VRMS	VINT
0.210	1.450	1.450
0.430	2.400	3.042
0.870	3.000	3.488
8.000	5.750	6.000

470

TIME	VRMS	VINT
0.270	1.450	1.450
0.730	2.400	2.812
1.400	3.100	3.715
8.000	5.602	6.000

640

TIME	VRMS	VINT
0.300	1.450	1.450
0.500	2.200	2.991
0.950	3.000	3.691
8.000	5.727	6.000

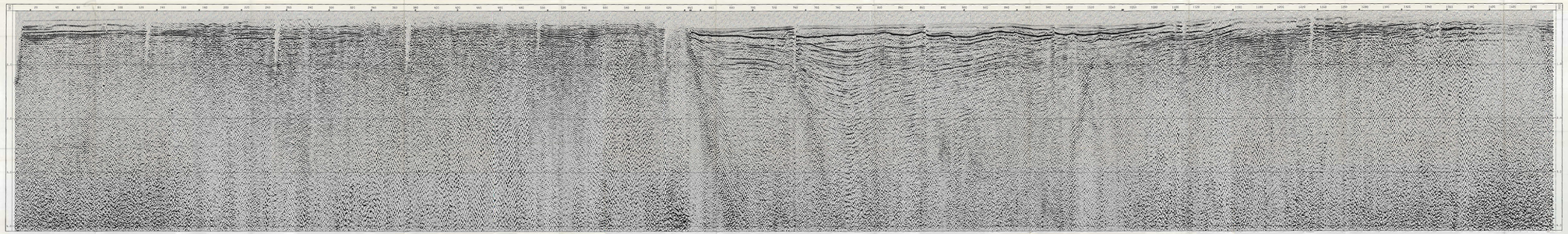
800

TIME	VRMS	VINT
0.300	1.450	1.450
1.070	2.600	2.928
2.550	4.300	5.193
8.000	5.515	6.000

DURHAM UNIVERSITY
SEISMIC PROCESSING
FARNELLA 1/81 LINE : 8

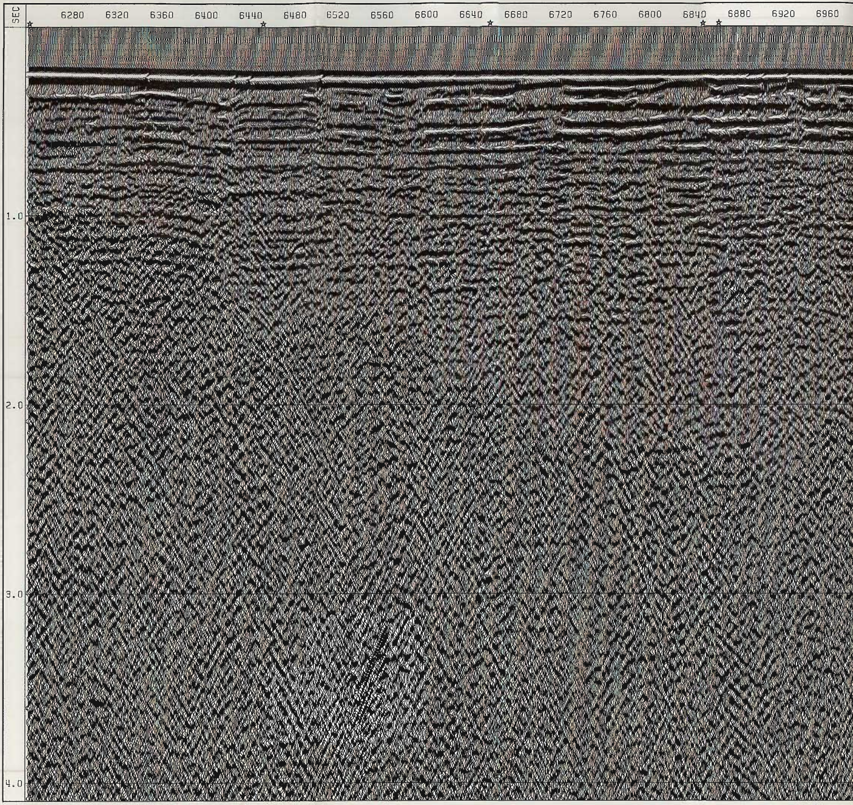
SYSTEM PARAMETERS

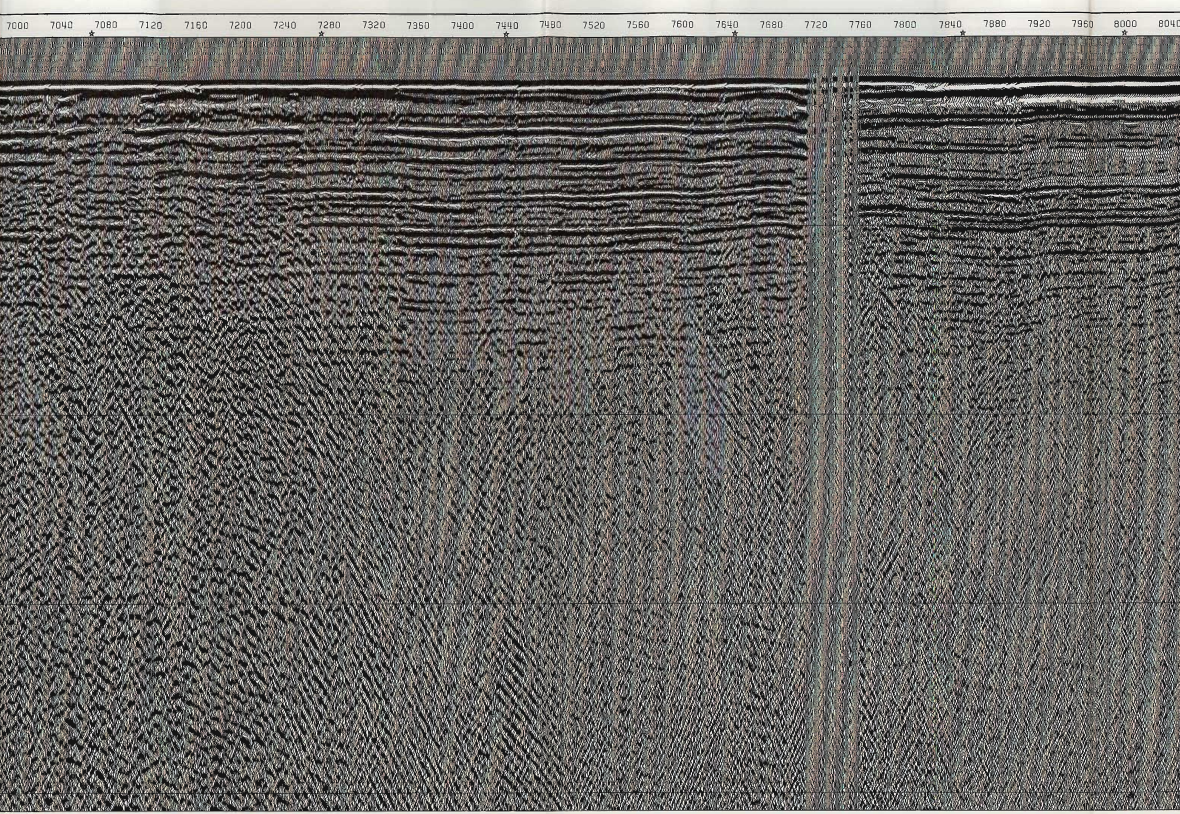
```
***FIELD DATA***
COLLECTED ON                FARNELLA CRUISE NO: 1/81
SOURCE/RECEIVER OFFSET      245 METRES
RECEIVER SPACING            100 METRES
SAMPLING INTERVAL           4 MSEC

***DEMULTIPLEX***

***SORT***
FOLD                        24
COMMON MID-POINT SPACING    50 METRES
RECORDER ERROR REMOVAL

***DBS***
TIME BREAK CORRECTION       64 MSEC
SOURCE DEPTH                10 METRES
RECEIVER DEPTH              10 METRES
POLARITY REVERSAL TRACES    2,5,10 AND 12
BAND-PASS FILTER            5,10 TO 40,60 HZ
VELOCITY FILTER             OPERATOR SIZE 11*11
                            FALSE VELOCITY STRUCTURE
                            TIME (MSEC)     RMS VELOCITY (KM/SEC)
                               400                1.4
                              1000                2.1
                              2000                3.0
MUTE                        TRACE NUMBER     TIME (MSEC)
                                 4                500
                                 7               1100
                                24               2800

***STACK***
RESAMPLED PRIOR TO NMO      1 MSEC
NMO AND STACK               24 FOLD

***DAS***
BAND-PASS FILTER            5,10 TO 40,60 HZ
GEOMETRIC SPEADING CORRECTION
DECONVOLUTION               PREDICTION ERROR
                            AUTOCORR. DESIGN         2000 MSEC FROM 0 MSEC
                            ACTIVE LENGTH             400 MSEC
                            PREDICTION LAG            100 MSEC
                            PRE-WHITENING               1 %
TRACE NORMALIZATION         UNIT ENERGY

***DISPLAY***
GAIN                        150
CLIP                        -0.1 TO 0.1 INCHES
TRACE SPACING               0.05 INCHES
VERTICAL SCALE              2.5 INCHES/SEC

PROCESSED BY                R.W.HOBBS
```

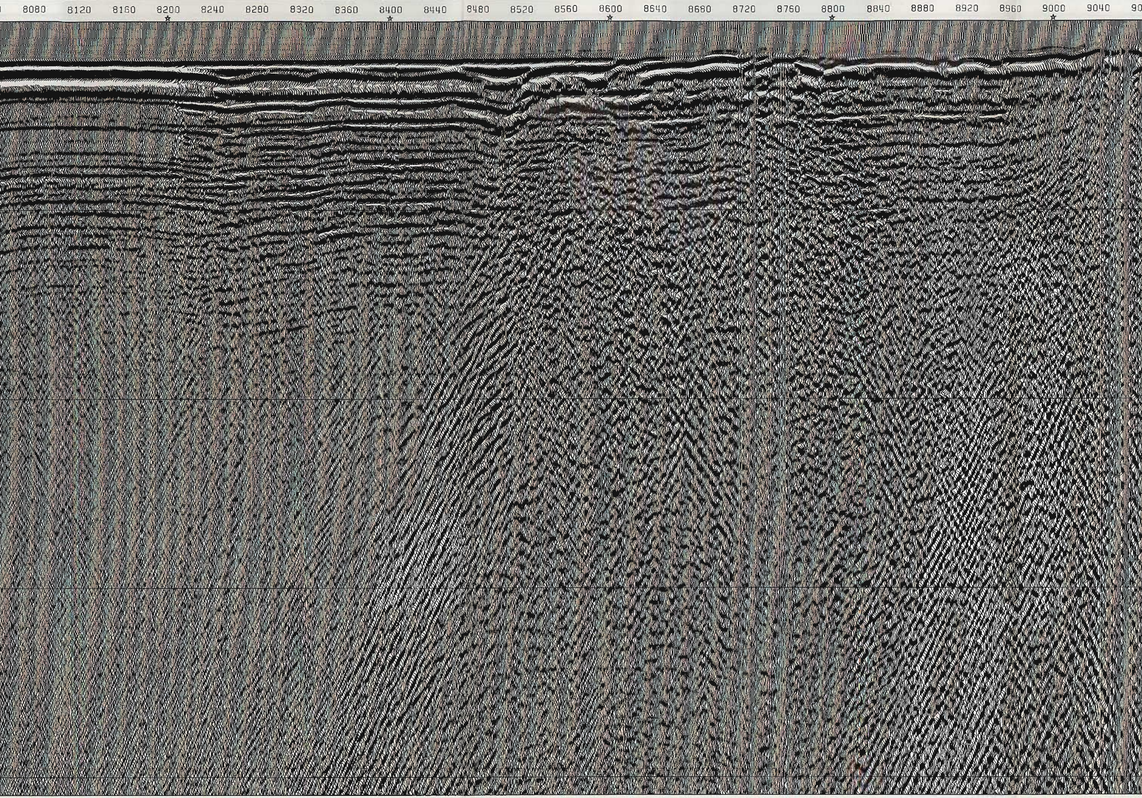


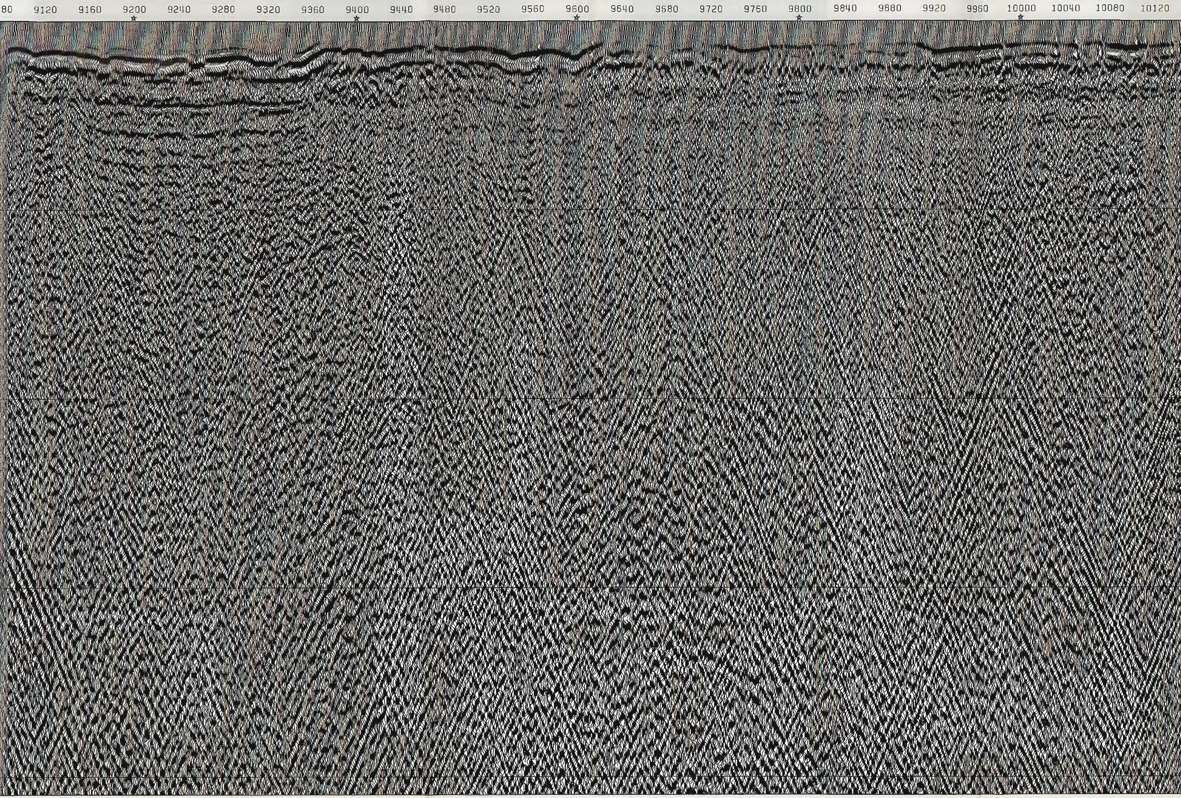
DURHAM UNIVER
SEISMIC PROCESS
FARNELLA 1/91 L
SYSTEM PARAMETERS

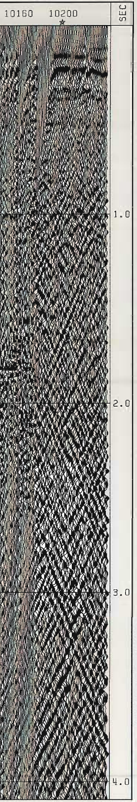
SEIS.EXP.INT. 1973   LINE : 11   START GATHER NO. 6241   END GATHER NO. 10240
COMMON MID-POINT   SECTION PLOT   POSITIVE LOBE SHADED
TRACE SPACING 0.015 INCHES   TRACE LENGTH 10.24 INCHES
PLOTTING GAIN 1.000E+02   LOW CLIP LEVEL -0.030 INCHES   HIGH CLIP LEVEL 0.030 INCHES











6242

TIME	VRMS	VINT
0.230	1.500	1.500
0.560	2.100	2.432
1.450	4.200	5.096
8.000	5.716	6.000

6862

TIME	VRMS	VINT
0.200	1.500	1.500
0.500	2.000	2.273
1.500	4.500	5.327
8.000	5.749	6.000

7648

TIME	VRMS	VINT
0.150	1.500	1.500
0.500	1.900	2.048
1.000	2.900	3.635
8.000	5.705	6.000

8400

TIME	VRMS	VINT
0.200	1.500	1.500
0.450	1.750	1.927
0.750	2.400	3.131
8.000	5.759	6.000

9200

TIME	VRMS	VINT
0.200	1.500	1.500
0.570	2.400	2.767
1.750	4.700	5.475
8.000	5.741	6.000

10000

TIME	VRMS	VINT
0.150	1.500	1.500
0.740	4.000	4.415
2.950	5.600	6.042
8.000	5.856	6.000

6452

TIME	VRMS	VINT
0.210	1.500	1.500
0.400	2.100	2.607
1.400	3.700	4.172
8.000	5.665	6.000

7088

TIME	VRMS	VINT
0.250	1.500	1.500
0.500	1.900	2.229
1.300	3.100	3.655
8.000	5.631	6.000

7852

TIME	VRMS	VINT
0.200	1.500	1.500
0.750	2.500	2.776
0.950	3.200	5.020
8.000	5.739	6.000

8600

TIME	VRMS	VINT
0.200	1.500	1.500
0.350	1.700	1.935
0.620	2.600	3.432
8.000	5.808	6.000

9400

TIME	VRMS	VINT
0.170	1.500	1.500
0.550	2.500	2.835
1.850	5.000	5.739
8.000	5.784	6.000

10200

TIME	VRMS	VINT
0.150	1.500	1.500
0.420	3.000	3.571
1.750	5.500	6.080
8.000	5.894	6.000

6658

TIME	VRMS	VINT
0.200	1.500	1.500
0.600	2.500	2.872
1.650	4.500	5.315
8.000	5.723	6.000

7274

TIME	VRMS	VINT
0.250	1.500	1.500
0.700	1.900	2.069
0.950	2.750	4.316
8.000	5.712	6.000

8000

TIME	VRMS	VINT
0.250	1.500	1.500
0.600	1.700	1.830
0.950	2.800	4.041
8.000	5.715	6.000

8800

TIME	VRMS	VINT
0.250	1.500	1.500
0.500	1.650	2.144
0.730	2.600	3.744
8.000	5.773	6.000

9600

TIME	VRMS	VINT
0.150	1.500	1.500
0.570	3.000	3.376
1.230	5.000	6.230
8.000	5.657	6.000

6846

TIME	VRMS	VINT
0.200	1.500	1.500
0.600	1.900	2.071
1.300	3.000	3.691
8.000	5.622	5.999

7440

TIME	VRMS	VINT
0.220	1.500	1.500
0.450	1.900	2.216
0.850	2.900	3.716
8.000	5.751	6.001

8200

TIME	VRMS	VINT
0.250	1.500	1.500
0.550	1.800	2.016
0.750	2.300	3.306
8.000	5.755	6.000

9000

TIME	VRMS	VINT
0.200	1.500	1.500
0.500	2.300	2.705
0.840	3.800	5.282
8.000	5.808	6.000

9800

TIME	VRMS	VINT
0.200	1.500	1.500
0.450	3.500	4.500
1.950	5.500	5.971
8.000	5.862	6.000

DURHAM UNIVERSITY
SEISMIC PROCESSING
SEIS.EXP.INT. 1973 LINE : 11

SYSTEM PARAMETERS

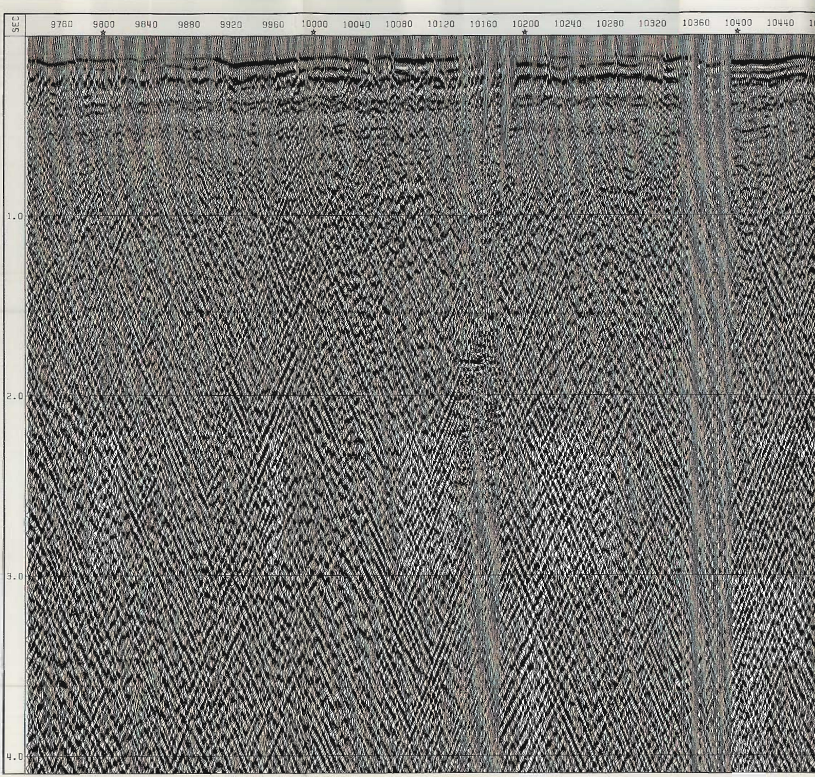
```
***FIELD DATA***
COLLECTED BY                  SEISMIC EXPLORATIONS INC.
YEAR                          1973
SHOT SPACING                  25 METRES
SOURCE/RECEIVER OFFSET        250 METRES
RECEIVER SPACING              25 METRES
SAMPLING INTERVAL             4 MSEC

***DEMULTIPLEX***

***SORT***
FOLD                          24
COMMON MID-POINT SPACING      12.5 METRES

***DBS***
BAND-PASS                     5,10 TO 30,80 HZ
TIME RAMP                     EXP(0.7*T)
DECONVOLUTION                 PREDICTION ERROR
                              AUTOCORR. DESIGN        4000 MSEC FROM 0 MSEC
                              ACTIVE LENGTH            400 MSEC
                              PREDICTION LAG            40 MSEC
                              PRE-WHITENING              1 %
REMOVE TIME RAMP
MUTE                          TRACE NUMBER    TIME (MSEC)
                                   2                0
                                  24             2200

***STACK***
RESAMPLED PRIOR TO NMO        2 MSEC
NMO AND STACK                 24 FOLD

***DAS***
TIME RAMP                     T(V**2)
DECONVOLUTION                 PREDICTION ERROR
                              AUTOCORR. DESIGN        2000 MSEC FROM 0 MSEC
                              ACTIVE LENGTH            400 MSEC
                              PREDICTION LAG           100 MSEC
                              PRE-WHITENING              1 %
TRACE NORMALIZATION           UNIT ENERGY

***DISPLAY***
GAIN                          100
CLIP                          -0.03 TO 0.03
TRACE SPACING                 0.015 INCHES
VERTICAL SCALE                1.25 INCHES/SEC

PROCESSED BY                  R.W.HOBBS
```

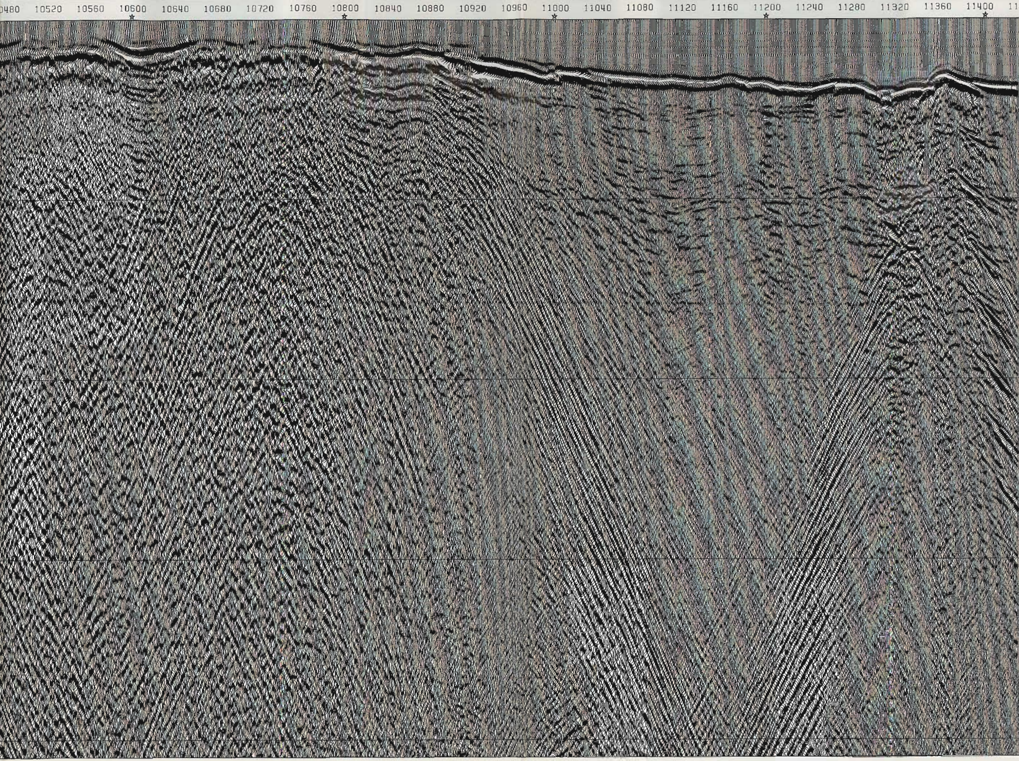
SEIS.EXP.INT. 1973 LINE : 11 START GATHER NO. 9729 END GATHER NO. 13728

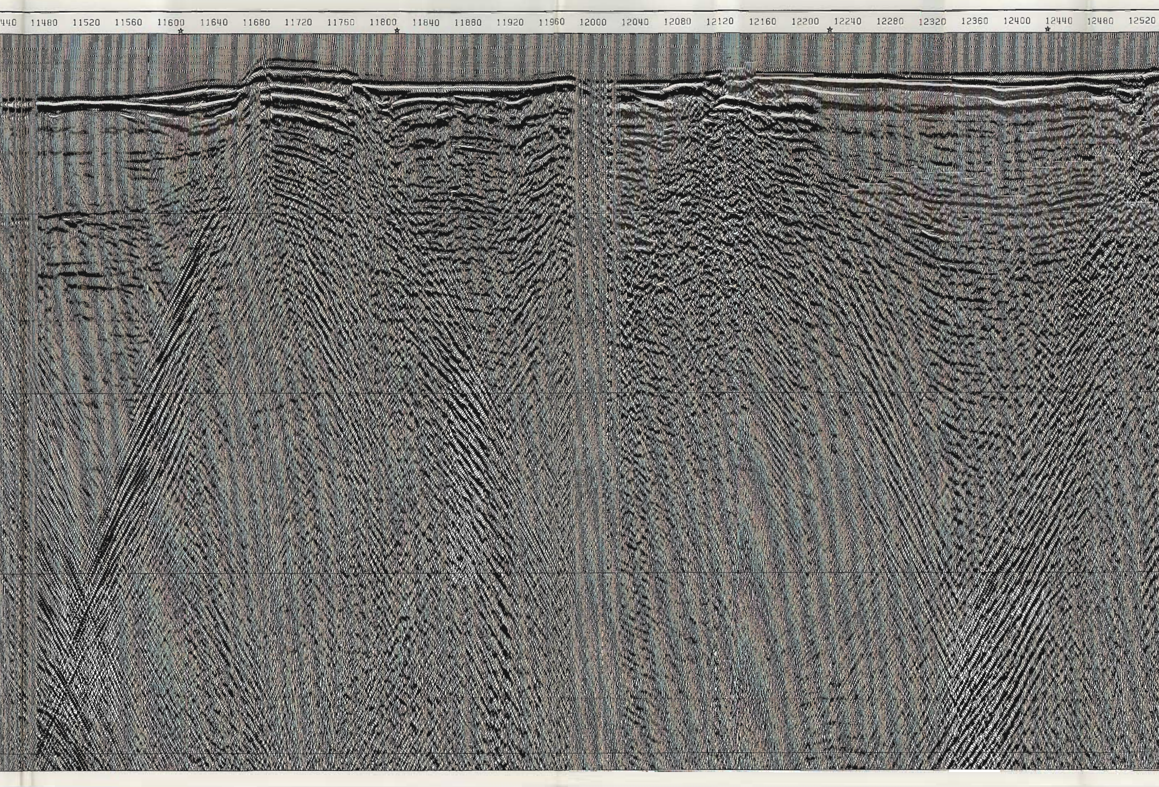
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED

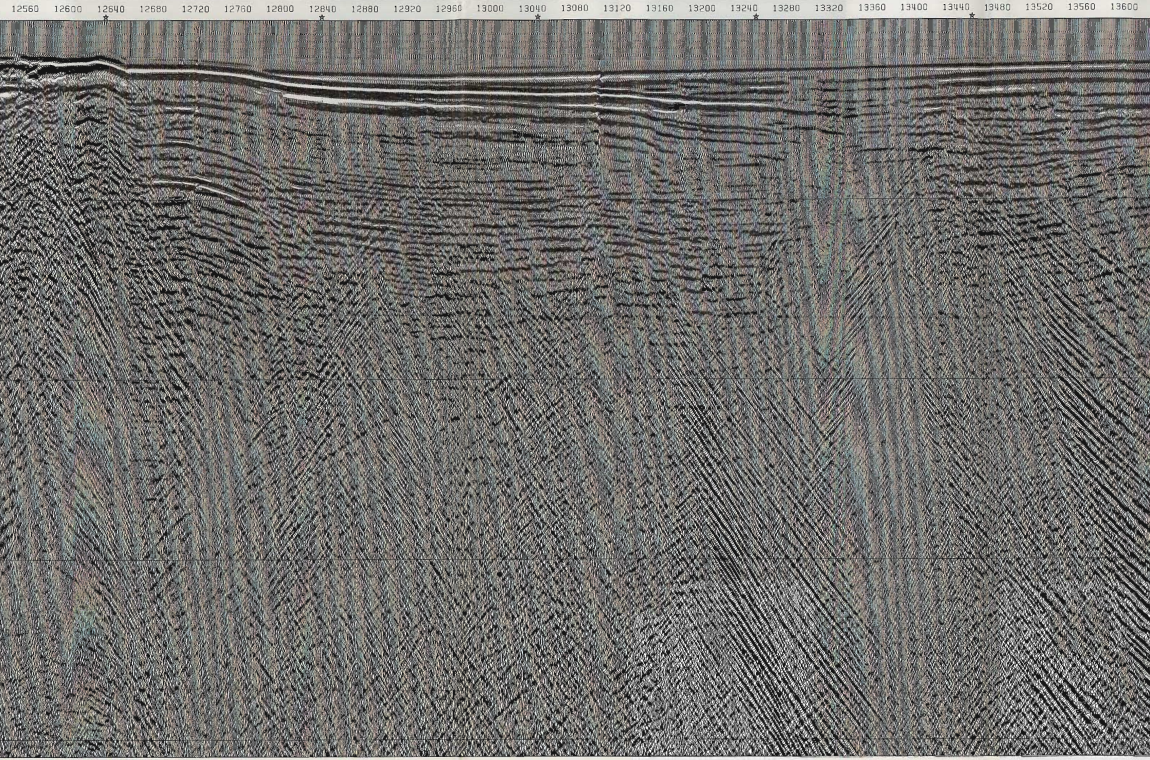
TRACE SPACING 0.015 INCHES TRACE LENGTH 10.24 INCHES

PLOTTING GAIN 1.000E+02 LOW CLIP LEVEL -0.030 INCHES HIGH CLIP LEVEL 0.030 INCHES









9800

TIME	VRMS	VINT
0.200	1.500	1.500
0.450	3.500	4.500
1.950	5.500	5.971
8.000	5.882	6.000

10600

TIME	VRMS	VINT
0.200	1.500	1.500
0.550	2.200	2.514
1.700	5.000	5.866
8.000	5.802	6.000

11406

TIME	VRMS	VINT
0.350	1.500	1.500
0.720	1.900	2.213
1.050	2.300	2.993
8.000	5.654	6.000

12430

TIME	VRMS	VINT
0.200	1.500	1.500
0.830	1.900	2.010
1.250	3.000	4.433
8.000	5.637	5.999

13252

TIME	VRMS	VINT
0.300	1.500	1.500
0.620	1.900	2.210
1.050	3.000	4.095
8.000	5.697	6.000

10000

TIME	VRMS	VINT
0.150	1.500	1.500
0.740	4.000	4.415
2.950	5.500	6.042
8.000	5.856	6.000

10800

TIME	VRMS	VINT
0.200	1.500	1.500
0.430	2.400	2.969
1.800	4.000	4.383
8.000	5.812	5.999

11610

TIME	VRMS	VINT
0.350	1.500	1.500
0.510	1.700	2.071
0.940	2.300	2.852
8.000	5.691	5.000

12636

TIME	VRMS	VINT
0.275	1.500	1.500
0.900	2.300	2.574
1.450	3.000	3.882
8.000	5.577	6.000

15456

TIME	VRMS	VINT
0.300	1.500	1.500
0.710	1.900	2.146
1.000	3.000	4.711
8.000	5.712	5.000

10200

TIME	VRMS	VINT
0.150	1.500	1.500
0.420	3.000	3.571
1.750	5.500	6.080
8.000	5.894	6.000

11000

TIME	VRMS	VINT
0.250	1.500	1.500
0.980	2.200	2.393
1.800	3.200	4.333
8.000	5.554	6.000

11814

TIME	VRMS	VINT
0.350	1.500	1.500
0.650	1.800	2.096
0.890	2.300	3.293
8.000	5.708	8.000

12840

TIME	VRMS	VINT
0.270	1.500	1.500
0.720	1.900	2.104
1.230	2.800	3.716
8.000	5.628	6.000

13662

TIME	VRMS	VINT
0.300	1.500	1.500
0.470	1.700	2.005
0.670	2.200	3.070
8.000	5.778	6.000

10400

TIME	VRMS	VINT
0.250	1.500	1.500
0.580	2.600	3.190
1.700	5.000	5.869
8.000	5.802	6.000

11200

TIME	VRMS	VINT
0.300	1.500	1.500
0.520	1.850	2.241
0.780	2.300	3.004
6.000	5.745	6.000

12224

TIME	VRMS	VINT
0.250	1.500	1.500
0.650	1.800	1.964
1.500	2.700	3.223
8.000	5.533	6.000

13048

TIME	VRMS	VINT
0.250	1.500	1.500
0.950	2.400	2.648
1.400	3.500	5.094
8.000	5.843	6.000

DURHAM UNIVERSITY

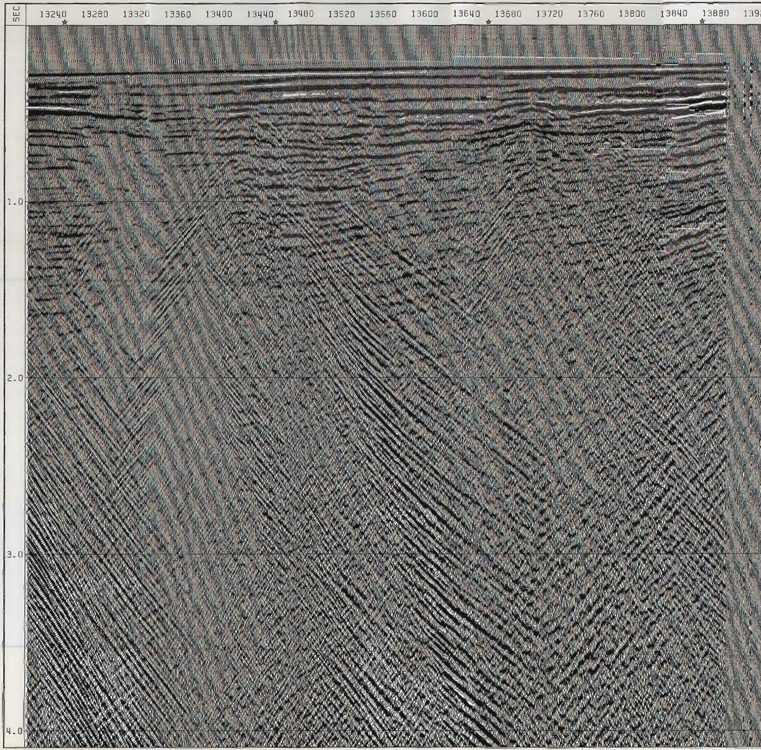
SEISMIC PROCESSING

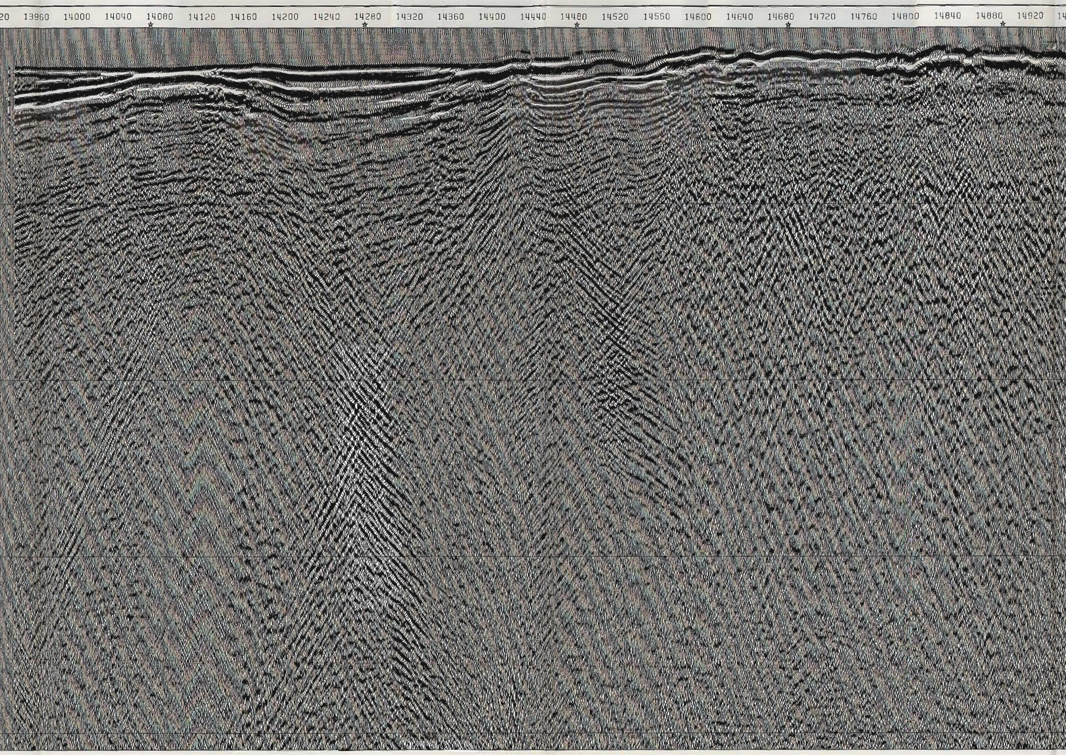
SEIS.EXP.INT. 1973 LINE : 11

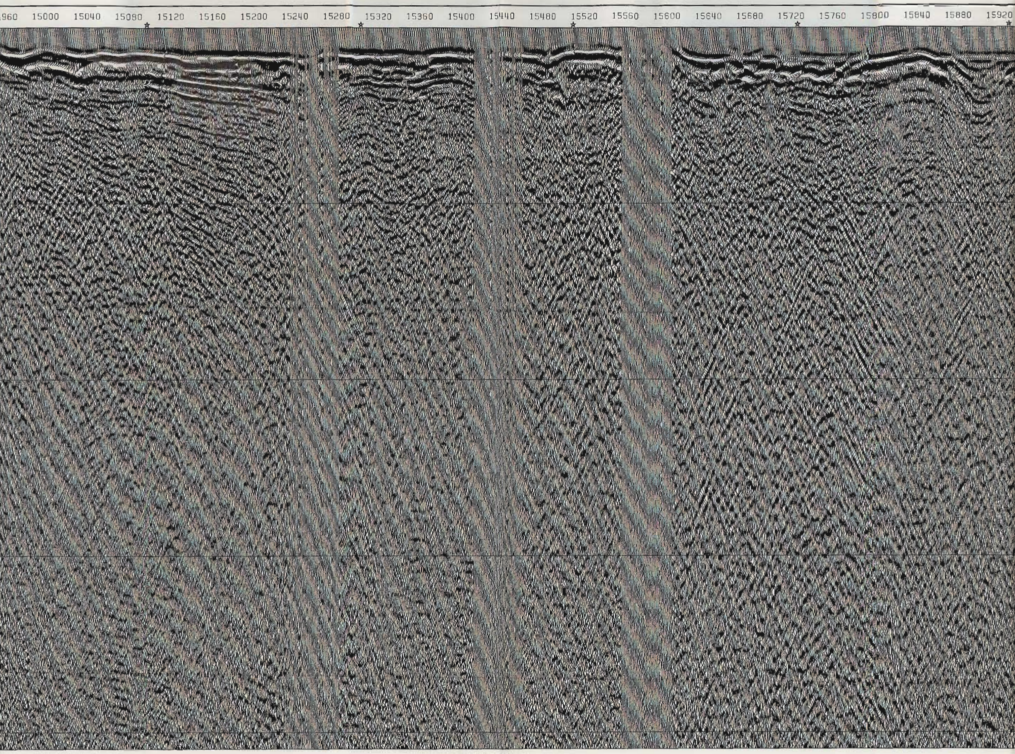
SYSTEM PARAMETERS

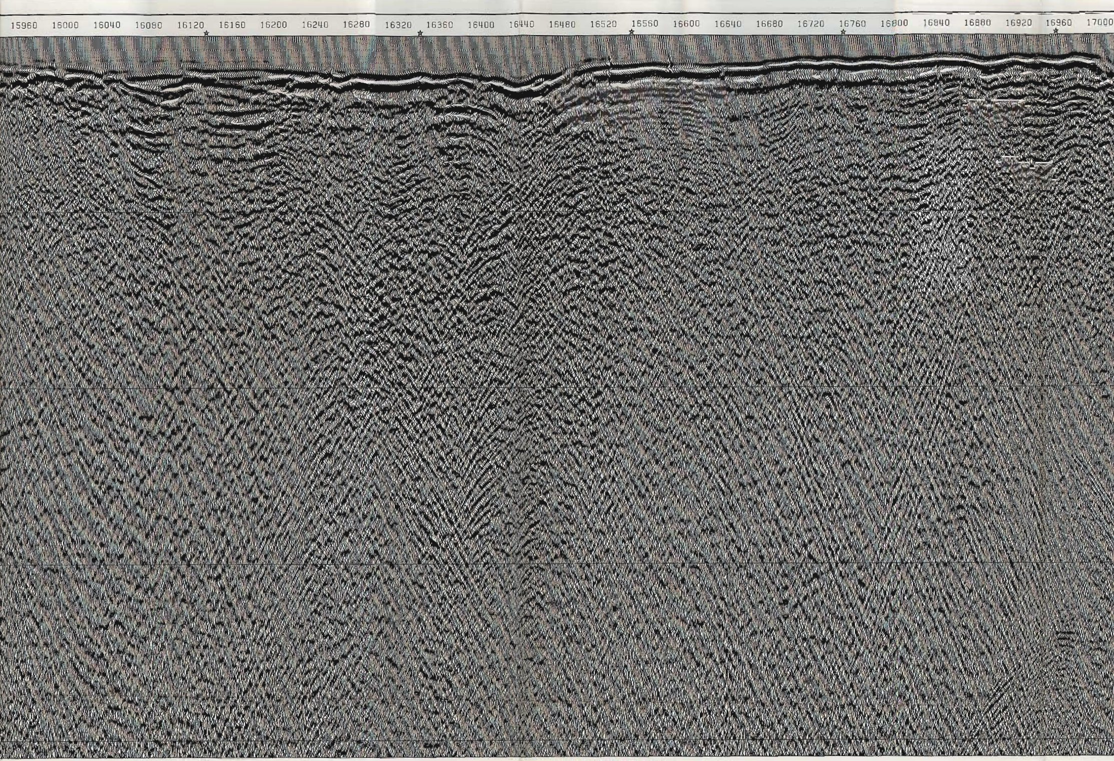
FIELD DATA		
COLLECTED BY	SEISMIC EXPLORATIONS INC.	
YEAR	1973	
SHOT SPACING	25 METRES	
SOURCE/RECEIVER OFFSET	250 METRES	
RECEIVER SPACING	25 METRES	
SAMPLING INTERVAL	4 MSEC	
DEMULTIPLEX		
SORT		
FOLD	24	
COMMON MID-POINT SPACING	12.5 METRES	
DBS		
BAND-PASS	5,10 TO 30,80 HZ	
TIME RAMP	EXP(0.7*T)	
DECONVOLUTION	PREDICTION ERROR	
	AUTOCORR. DESIGN	4000 MSEC FROM 0 MSEC
	ACTIVE LENGTH	400 MSEC
	PREDICTION LAG	40 MSEC
	PRE-WHITENING	1 %
REMOVE TIME RAMP		
MUTE	TRACE NUMBER	TIME (MSEC)
	2	0
	24	2200
STACK		
RESAMPLED PRIOR TO NMO	2 MSEC	
NMO AND STACK	24 FOLD	
DAS		
TIME RAMP	T(V**2)	
DECONVOLUTION	PREDICTION ERROR	
	AUTOCORR. DESIGN	2000 MSEC FROM 0 MSEC
	ACTIVE LENGTH	400 MSEC
	PREDICTION LAG	100 MSEC
	PRE-WHITENING	1 %
TRACE NORMALIZATION	UNIT ENERGY	
DISPLAY		
GAIN	100	
CLIP	-0.03 TO 0.03	
TRACE SPACING	0.015 INCHES	
VERTICAL SCALE	1.25 INCHES/SEC	
PROCESSED BY	R.W.HOBBS	

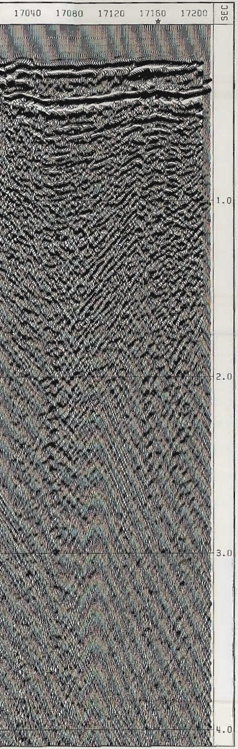
SEIS.EXP.INT. 1973 LINE : 11 START GATHER NO. 13217 END GATHER NO. 17216
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.015 INCHES TRACE LENGTH 10.24 INCHES
PLOTTING GAIN 1.000E+02 LOW CLIP LEVEL -0.030 INCHES HIGH CLIP LEVEL 0.030 INCHES











13252

TIME	VRMS	VINT
0.300	1.500	1.500
0.620	1.900	2.210
1.050	3.000	4.095
6.000	5.697	6.000

14072

TIME	VRMS	VINT
0.200	1.500	1.500
0.300	1.600	1.783
0.850	2.500	2.874
8.000	5.731	6.000

14894

TIME	VRMS	VINT
0.200	1.500	1.500
0.700	2.700	3.051
1.300	3.500	4.247
8.000	5.669	6.000

15727

TIME	VRMS	VINT
0.200	1.500	1.500
0.800	2.300	2.511
1.750	3.500	4.256
8.000	5.550	6.000

16548

TIME	VRMS	VINT
0.200	1.500	1.500
0.700	2.600	2.926
1.200	3.800	5.019
0.000	5.724	6.000

13456

TIME	VRMS	VINT
0.300	1.500	1.500
0.710	1.900	2.146
1.000	3.000	4.711
6.000	5.712	6.000

14278

TIME	VRMS	VINT
0.200	1.500	1.500
0.350	1.600	1.724
1.250	3.300	3.759
8.000	5.634	6.000

15096

TIME	VRMS	VINT
0.200	1.500	1.500
0.700	2.500	2.802
1.220	3.100	3.759
6.000	5.655	6.000

15930

TIME	VRMS	VINT
0.200	1.500	1.500
0.850	3.200	3.563
1.550	4.500	5.693
8.000	5.740	6.000

16752

TIME	VRMS	VINT
0.200	1.500	1.500
0.650	2.300	2.577
1.500	4.200	5.204
8.000	5.706	6.000

13662

TIME	VRMS	VINT
0.300	1.500	1.500
0.470	1.700	2.005
0.670	2.200	3.070
8.000	5.778	6.000

14464

TIME	VRMS	VINT
0.200	1.500	1.500
0.420	1.900	2.201
0.370	2.600	3.114
8.000	5.729	6.000

15304

TIME	VRMS	VINT
0.200	1.500	1.500
0.950	2.800	3.055
1.400	3.400	4.406
8.000	5.632	6.000

16136

TIME	VRMS	VINT
0.200	1.500	1.500
0.700	2.600	2.926
1.700	4.000	4.740
8.000	5.635	6.000

16958

TIME	VRMS	VINT
0.200	1.500	1.500
0.550	1.850	2.023
1.310	4.100	5.148
8.000	5.732	6.000

13868

TIME	VRMS	VINT
0.300	1.500	1.500
0.450	1.650	1.915
1.250	3.000	3.540
8.000	5.637	5.999

14688

TIME	VRM3	VINT
0.200	1.500	1.500
0.430	2.100	2.508
0.600	2.900	4.304
8.000	5.825	6.000

15510

TIME	VRMS	VINT
0.200	1.500	1.500
0.800	3.000	3.354
1.250	3.500	4.246
8.000	5.682	6.000

16342

TIME	VRMS	VINT
0.200	1.500	1.500
0.750	2.500	2.776
1.300	4.000	5.413
8.000	5.723	6.000

17186

TIME	VRM3	VINT
0.200	1.500	1.500
0.700	2.100	2.297
1.510	4.000	5.101
8.000	5.877	6.000

DURHAM UNIVERSITY

SEISMIC PROCESSING

SEIS.EXP.INT. 1973 LINE : 11

SYSTEM PARAMETERS

```
***FIELD DATA***
COLLECTED BY                SEISMIC EXPLORATIONS INC.
YEAR                        1973
SHOT SPACING                25 METRES
SOURCE/RECEIVER OFFSET      250 METRES
RECEIVER SPACING            25 METRES
SAMPLING INTERVAL           4 MSEC

***DEMULTIPLEX***

***SORT***
FOLD                        24
COMMON MID-POINT SPACING    12.5 METRES

***DBS***
BAND-PASS                   5,10 TO 30,80 HZ
TIME RAMP                   EXP(0.7*T)
DECONVOLUTION               PREDICTION ERROR
                            AUTOCORR. DESIGN        4000 MSEC FROM 0 MSEC
                            ACTIVE LENGTH            400 MSEC
                            PREDICTION LAG            40 MSEC
                            PRE-WHITENING              1 %
REMOVE TIME RAMP
MUTE                        TRACE NUMBER    TIME (MSEC)
                                 2               0
                                24            2200

***STACK***
RESAMPLED PRIOR TO NMO      2 MSEC
NMO AND STACK               24 FOLD

***DAS***
TIME RAMP                   T(V**2)
DECONVOLUTION               PREDICTION ERROR
                            AUTOCORR. DESIGN        2000 MSEC FROM 0 MSEC
                            ACTIVE LENGTH            400 MSEC
                            PREDICTION LAG           100 MSEC
                            PRE-WHITENING              1 %
TRACE NORMALIZATION         UNIT ENERGY

***DISPLAY***
GAIN                        100
CLIP                        -0.03 TO 0.03
TRACE SPACING               0.015 INCHES
VERTICAL SCALE              1.25 INCHES/SEC

PROCESSED BY                R.W.HOBBS
```

SEA OF THE HEBRIDES REGION

TRANSVERSE MERCATOR PROJECTION

SCALE 1:250000

COMBINED CHART FOR REFLECTION AND REFRACTION DATA SURVEYS

WINCH	British Institutions Reflection Profiling Syndicate 1982
F1/81	Farnella survey 1981
SEI	Seismic Exploration Int. survey 1973
IGS	Institute of Geological Sciences survey 1969
★ SHOTS	Western Isles Seismic (refraction) Experiment 1979 WISE
• STATIONS	

8°W 7°W 6°W

57°N 56°30' 56°N

BARRA RHUM SLEAT COLL TIREE MULL JURA

WINCH 2 (S4) WINCH 2 (S5)

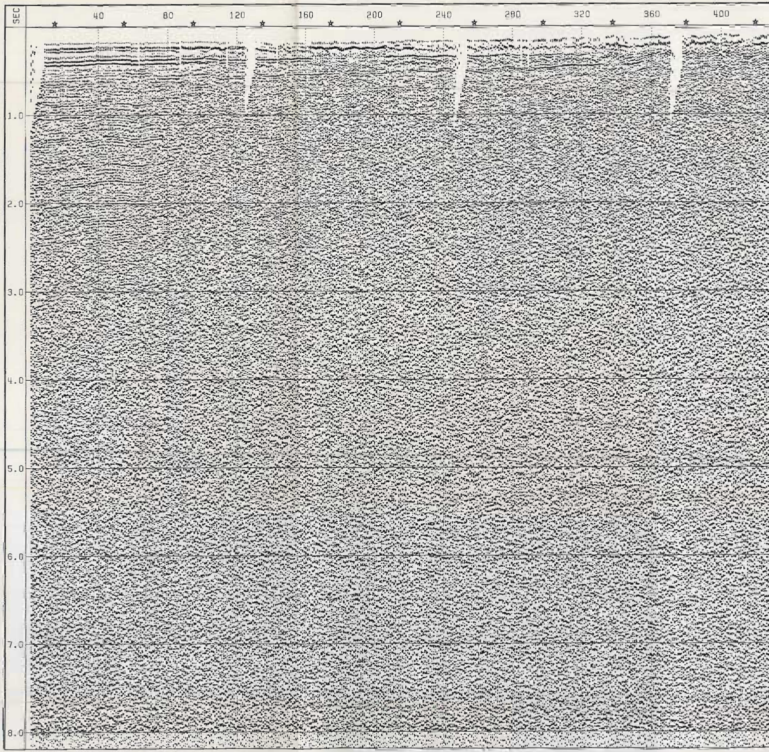
SEI 11 7100 8700 10300 11900 13500 15100 16700 SEI 11

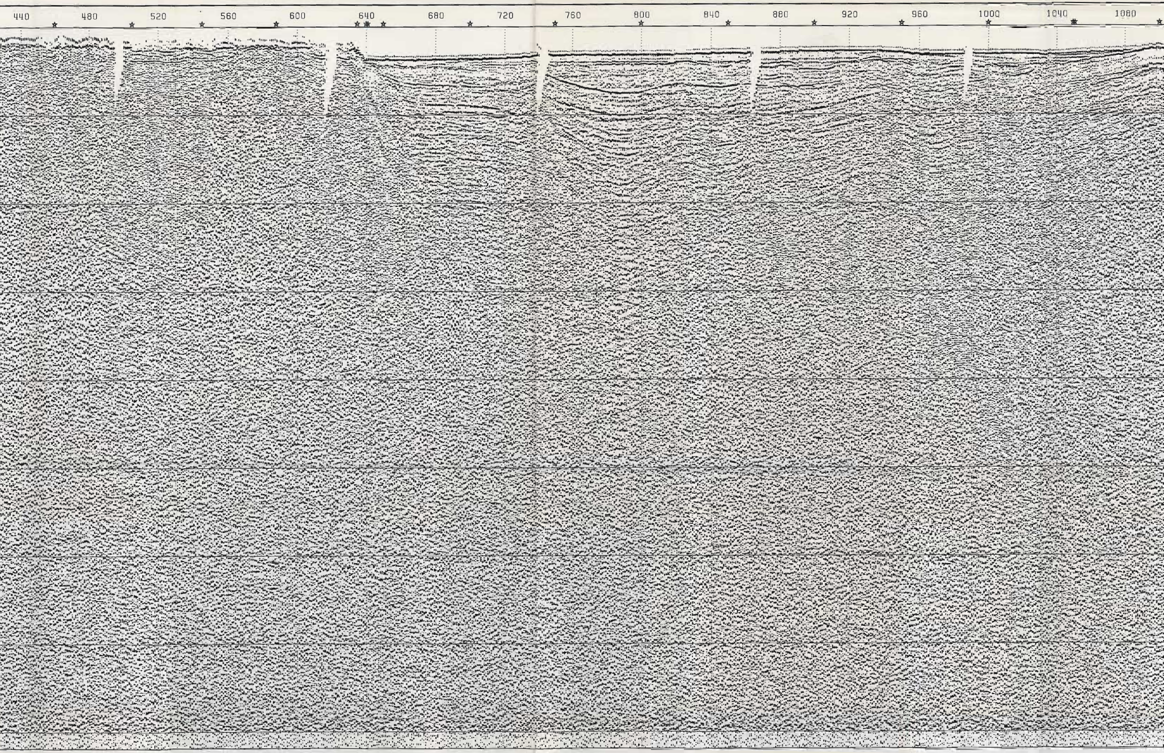
F1/81.1 F1/81.2 F1/81.3 F1/81.4 F1/81.6 F1/81.7 F1/81.8 F1/81.9 F1/81.10

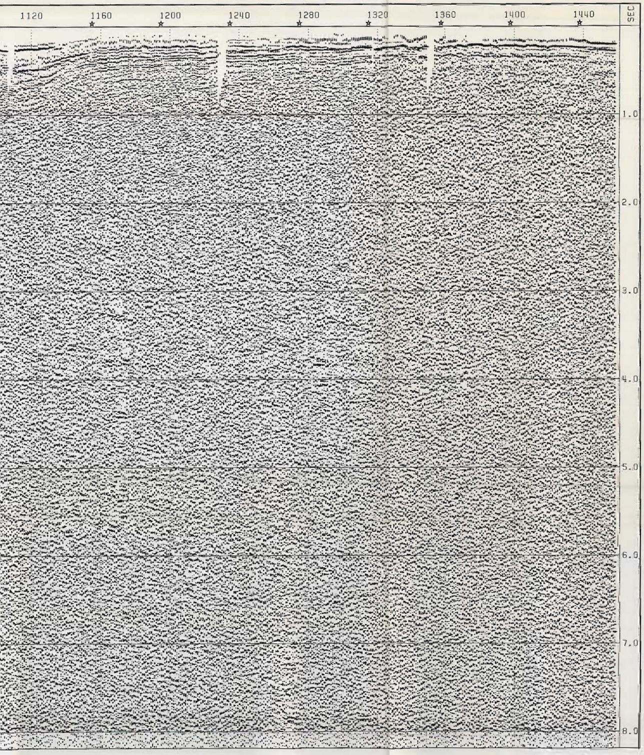
IGS 1 IGS 2 IGS 3 IGS 4 IGS 8 IGS 11 IGS 12 IGS 13

★19 ★20 ★21 ★22 ★23

FARNELLA 1/81 LINE : 7 START GATHER NO. 1 END GATHER NO. 1458
COMMON MID-POINT SECTION PLOT POSITIVE LOBE SHADED
TRACE SPACING 0.025 INCHES TRACE LENGTH 10.24 INCHES 50 SAMPLE AGC APPLIED
PLOTTING GAIN 2.500E+00 LOW CLIP LEVEL 0.005 INCHES HIGH CLIP LEVEL 0.050 INCHES







15

TIME	VRMS	VINT
0.200	1.450	1.450
0.350	2.700	3.769
2.000	3.805	4.000
4.000	5.533	6.837

55

TIME	VRMS	VINT
0.190	1.450	1.450
0.560	2.700	3.155
1.550	3.500	3.880
4.000	5.603	6.596

95

TIME	VRMS	VINT
0.180	1.450	1.450
0.320	2.400	3.235
0.950	3.500	3.943
4.000	5.760	6.300

135

TIME	VRMS	VINT
0.170	1.450	1.450
0.320	2.500	3.309
0.900	3.200	3.527
4.000	5.753	6.303

175

TIME	VRMS	VINT
0.160	1.450	1.450
0.350	3.000	3.648
1.000	4.405	5.001
4.000	5.825	6.227

215

TIME	VRMS	VINT
0.150	1.450	1.450
0.320	2.500	3.148
2.000	4.690	5.000
4.000	5.701	6.558

258

TIME	VRMS	VINT
0.200	1.456	1.450
0.340	2.300	3.137
1.000	3.515	3.999
4.000	5.746	6.319

298

TIME	VRMS	VINT
0.160	1.450	1.450
0.310	2.300	2.946
0.820	3.600	4.198
4.000	5.800	6.243

338

TIME	VRMS	VINT
0.150	1.450	1.450
0.300	3.000	3.987
1.000	4.494	4.999
4.000	5.833	6.216

380

TIME	VRMS	VINT
0.180	1.450	1.450
0.420	2.900	3.625
1.000	4.246	4.999
4.000	5.810	6.245

420

TIME	VRMS	VINT
0.120	1.450	1.450
0.350	2.400	2.769
1.000	4.274	5.000
4.000	5.812	6.241

460

TIME	VRMS	VINT
0.120	1.450	1.450
0.460	3.300	3.741
0.750	4.000	4.910
4.000	5.842	6.190

505

TIME	VRMS	VINT
0.110	1.450	1.450
0.320	2.600	3.033
0.750	4.000	4.783
4.000	5.842	6.190

545

TIME	VRMS	VINT
0.100	1.450	1.450
0.330	2.800	3.215
0.820	4.400	5.208
4.000	5.858	6.176

586

TIME	VRMS	VINT
0.100	1.450	1.450
0.300	3.000	3.528
0.550	3.600	4.209
4.000	5.867	6.152

635

TIME	VRMS	VINT
0.150	1.450	1.450
0.320	2.500	3.148
0.900	3.900	4.489
4.000	5.802	6.247

640

TIME	VRMS	VINT
0.350	1.450	1.450
0.500	3.248	5.501
0.700	4.022	5.499
4.000	5.729	6.029

641

TIME	VRMS	VINT
0.350	1.450	1.450
0.400	1.530	2.002
0.500	1.916	2.999
0.700	3.125	5.000
4.000	5.604	6.000

650

TIME	VRMS	VINT
0.350	1.450	1.450
0.650	1.700	1.952
0.850	2.150	3.202
1.200	3.300	5.110
4.000	5.335	5.999

700

TIME	VRMS	VINT
0.350	1.450	1.450
0.450	1.550	1.858
0.650	1.850	2.391
1.050	2.550	3.392
4.000	5.316	6.000

750

TIME	VRMS	VINT
0.300	1.450	1.450
0.450	1.600	1.864
0.850	2.000	2.371
1.250	2.650	3.667
4.000	5.191	6.000

800

TIME	VRMS	VINT
0.300	1.450	1.450
0.400	1.550	1.817
0.850	2.250	2.725
1.500	2.950	3.859
4.000	5.076	6.000

850

TIME	VRMS	VINT
0.250	1.450	1.450
0.450	1.550	1.667
0.700	2.000	2.622
1.600	3.200	3.885
4.000	5.069	6.000

900

TIME	VRMS	VINT
0.250	1.450	1.450
0.500	1.800	1.737
0.850	2.100	2.656
1.500	3.550	4.829
4.000	5.218	6.000

950

TIME	VRMS	VINT
0.250	1.450	1.450
0.450	1.550	1.667
0.800	1.900	2.689
1.350	3.250	4.016
4.000	5.236	6.000

1000

TIME	VRMS	VINT
0.250	1.450	1.450
0.450	1.550	1.667
0.550	1.814	2.699
1.050	3.000	3.909
4.000	5.377	6.000

1050

TIME	VRMS	VINT
0.250	1.450	1.450
0.450	1.550	1.667
0.475	1.631	2.701
0.650	2.500	3.999
4.000	5.583	6.000

1051

TIME	VRMS	VINT
0.250	1.450	1.450
0.450	1.550	1.667
0.650	2.500	3.861
4.000	5.583	6.000

1100

TIME	VRMS	VINT
0.200	1.450	1.450
0.225	1.521	2.000
0.500	3.137	4.000
4.000	5.721	6.000

1275

TIME	VRMS	VINT
0.150	1.450	1.450
0.270	2.300	3.045
0.820	3.200	3.749
4.000	5.831	6.193

1438

TIME	VRMS	VINT
0.100	1.450	1.450
0.300	3.000	3.528
1.000	3.728	4.000
4.000	5.765	5.299

1155

TIME	VRMS	VINT
0.180	1.450	1.450
0.300	2.500	3.531
1.000	3.616	4.000
4.000	5.753	6.310

1315

TIME	VRMS	VINT
0.120	1.450	1.450
0.260	2.500	3.131
1.000	3.569	3.999
4.000	5.760	6.305

1195

TIME	VRMS	VINT
0.150	1.450	1.450
0.420	2.600	3.057
1.000	3.481	4.000
4.000	5.746	6.323

1358

TIME	VRMS	VINT
0.110	1.450	1.450
0.300	2.400	2.807
1.000	3.596	4.001
4.000	5.755	6.313

1235

TIME	VRMS	VINT
0.150	1.450	1.450
0.370	2.100	2.446
0.720	3.000	3.722
4.000	5.794	6.242

1398

TIME	VRMS	VINT
0.100	1.450	1.450
0.420	3.500	3.927
1.000	3.798	4.000
4.000	5.771	6.293

DURHAM UNIVERSITY

SEISMIC PROCESSING

FARNELLA 1/81 LINE : 7

SYSTEM PARAMETERS

```
***FIELD DATA***
COLLECTED ON                 FARNELLA CRUISE NO: 1/81
SOURCE/RECEIVER OFFSET       245 METRES
RECEIVER SPACING             100 METRES
SAMPLING INTERVAL            4 MSEC

***DEMULTIPLEX***

***SORT***
FOLD                         24
COMMON MID-POINT SPACING     50 METRES
RECORDER ERROR REMOVAL

***DBS***
TIME BREAK CORRECTION        64 MSEC
SOURCE DEPTH                 10 METRES
RECEIVER DEPTH               10 METRES
POLARITY REVERSAL TRACES     2,5,10 AND 12
BAND-PASS FILTER             5,10 TO 40,60 HZ
TIME RAMP                    EXP( T )
DECONVOLUTION                PREDICTION ERROR
                             AUTOCORR. DESIGN          4000 MSEC FROM 0 MSEC
                             ACTIVE LENGTH              400 MSEC
                             PREDICTION LAG              40 MSEC
                             PRE-WHITENING                1 %
REMOVE TIME RAMP
MUTE                         TRACE NUMBER      TIME (MSEC)
                                   2                500
                                  24               2700

***STACK***
RESAMPLED PRIOR TO NMO       2 MSEC
NMO AND STACK                24 FOLD

***DAS***
BAND-PASS FILTER             5,10 TO 40,60 HZ
GEOMETRIC SPEADING CORRECTION
DECONVOLUTION                PREDICTION ERROR
                             AUTOCORR. DESIGN          2000 MSEC FROM 0 MSEC
                             ACTIVE LENGTH              400 MSEC
                             PREDICTION LAG             100 MSEC
                             PRE-WHITENING                1 %
TRACE NORMALIZATION          UNIT ENERGY

***DISPLAY***
A.G.C. APPLIED               200 MSEC WINDOW
GAIN                         2.5
CLIP                         0.005 TO 0.05 INCHES
TRACE SPACING                0.025 INCHES
VERTICAL SCALE               1.25 INCHES/SEC

PROCESSED BY                 R.W.HOBBS
```