Suggested Basic Processing Procedures for UTIG OBS Data from Artificial Sources

by

Yosio Nakamura

January 3, 1994

Institute for Geophysics
The University of Texas at Austin
7801 North MoPac Expressway
Austin, Texas 78759-8397

University of Texas Institute for Geophysics Technical Report No. 129

TABLE OF CONTENTS

INTRODUCTION

There are no firmly fixed procedures for processing seismic data acquired using ocean-bottom seismographs (OBS) developed by the Institute for Geophysics of the University of Texas at Austin (UTIG). As the instrumentation and the mode of data acquisition are improved to adopt to advancing technology and special needs of experiments, there always are needs for revised and improved processing procedures. Yet, it is often useful to have a suggested basic procedure to follow, even though each processing of specific data set may deviate significantly from such a set procedure.

This report is an attempt to provide such a procedure. It is specifically for the recently upgraded UTIG OBS [*Nakamura and Garmany*, 1991]. It describes basic procedures we have been following up to making SEG-Y format tapes. Once standard format tapes are made, further processing of the data can be done by using any of the many processing program packages available in market.

In this report, it is assumed that most processing is done on a SUN computer. The programs mentioned here are FORTRAN programs written for a SUN except where specifically stated otherwise. Equivalent programs in QuickBasic for a Macintosh are also available in many cases. Listings of most, but not all, of the programs mentioned here are given in Appendices D and E to assist those who need to know exactly what is being done with these programs. Copies of the program files are available from the author.

The procedures and programs given here do not go beyond being a general guide. The user is encouraged to modify them as appropriate to respond to specific needs of his/her experiment.

RAW DATA

The following data are normally needed to process the acquired OBS data properly. The first group of five constitutes the primary data:

1. Raw OBS data on 1/4" cartridge tapes
2. OBS start-up/clock-calibration capture files
3. Shot-time logs
4. Navigation logs
5. Bathymetry logs

In addition, we also need the following information:

6. OBS deployment and recovery times, coordinates and water depths
7. Source setback (distance from navigation receiver antenna to sound source)
8. Source depth
9. Echo sounder depth (normally, the ship's draft) and time-to-depth conversion factor (sound speed) used
10. Shot delay (time interval from logged shot time to actual gun firing), if known

SUGGESTED PROCESSING PROCEDURES

Pre-Processing

1. Raw OBS data: The raw OBS data may require certain corrections. Most Tandberg drives we now use record data which are shifted by one byte for a certain yet unknown reason, and thus the data must be shifted back. Other data irregularities may occur, such as unreadable data blocks and clock errors. A quick QC immediately following each recovery of an instrument may tell us if any corrections are needed. A suggested procedure is:

(a) to read data into a disk storage,
(b) to correct the byte shift, if necessary,
(c) to obtain a header listing to see if there are any data irregularities,
(d) to make corrections, if necessary, and
(e) to copy the corrected data to another recording medium, such as a 9-track tape or an Exabyte tape for use in further processing.

Unix 'dd' command may be used to read the data from a cartridge tape to a SUN disk file. Although cartridge tapes are written in 512-byte blocks, our OBS uses 4K-byte blocks (i.e., 8 tape blocks per OBS block; c.f., Appendices A and B). Thus it is suggested that a block size (bs=) that is a multiple of 4K and an integer fraction (1/1, 1/2, 1/3, etc.) of the OBS record size (maximum 512K) be used for the 'dd' transfer. One may find that some defective tapes cannot be read past a bad spot on a SUN using this procedure. In those cases, application **Qutape** on Macintosh may be used to read the data.

The byte shift error may be corrected by running a program **datacorrc44**. The data file must be named **obsdata.ext**, where 'ext' is a file name extension of up to 8 characters. The correction is done in place without generating a new file.

Header listings may be made by using a program **HeaderListL**. This program reads the headers of OBS data stored in a file named **obsdata.ext** and produces a header list of the last block of each record in hexadecimal format in an output file named **HdrList.ext**. (The reason why we list the last, but not the first, headers is that only the last headers contain data needed later for a sampling-rate calibration. See Appendix A.) Optionally, program **HeaderListA** may be used to list the headers of all of the OBS blocks. Program **HdrTimInt** converts the hexadecimal time in **HdrList.ext** into decimal time and computes the time increment between blocks and the time spent transferring the data from OBS memory to tape, storing the output list in a file named **HdrInt.ext**. Other optional programs include **HeaderListLp** and **HeaderListAp**, which are identical to **HeaderListL** and **HeaderListA**, respectively, except that they use pre-byte-shift-corrected data as inputs.

2. Clock calibration data: OBS clock calibration data must be extracted from the OBS start-up/clock-calibration capture files. This may be done on a Macintosh using a program **ClockCalSelect**. The program extracts all the clock calibration data for a given OBS station ID number from the capture file.

3. Shot time log: The shot-time log may need some clean up, such as to remove extraneous characters, and sometimes to fill in missing shots and to remove those shots which are not actually fired. Program **ShotTimeCheck** may be used on a Macintosh to detect irregularities in shot intervals. You may need to refer to watch-stander's logs

and other available data, such as bathymetry charts, which may show indications of shots actually fired.

It is often convenient to have a shot number assigned to each shot. Program **AssShotNos** may be used to assign shot numbers to shot times in a file named **ShotTimes.ext** to produce a file named **ShotNT.ext**, where '**ext**' again is a file name extension of up to 8 characters.

4. Navigation data: Ship's location, in geographic coordinates, need be extracted, or sometimes recomputed, from the navigation log. The procedure depends on the type of navigation used and the format of the navigation log. The resulting navigation file should contain the time and the coordinates. The coordinates are either in latitude-longitude, easting-northing in UTM (Universal Transverse Mercator) projection, or Loran-C TD's.

5. Bathymetry data: The bathymetry logs, if manually read from an echo sounder chart, need be entered into computer files. Bathymetric readings every five minutes will be satisfactory. It is advisable to check the correctness of the bathymetry file entries because keyboard entry errors are rather common. Plotting of bathymetric profiles on a computer screen using any available plotting software, such as **Excel Chart** on a Macintosh, is an effective way to detect gloss keyboard entry errors.

Preliminary Clock Correction

Preliminary OBS clock corrections applicable to the time period of data acquisition may be estimated based on the following initially-available data:

(a) clock drift rate immediately preceding the deployment,
(b) the last clock calibration before deployment,
(c) time of deployment,
(d) variation in clock drift rate between the sleeping-on-the-sea-floor (cold temperature) and data-acquisition (slightly warmer temperature) phases estimated from the temperature coefficient of clock crystal frequency as measured in an environmental chamber and estimated temperature rise for the data acquisition rate (sampling rate and number of channels) being used,
(e) time of the start of data acquisition,
(f) time of the end of data acquisition, and
(g) the clock calibration immediately following recovery.

The estimation procedure is as follows:

(1) Let t_1 be the time of a clock calibration about two ours before deployment, t_2 be the time of the last clock calibration before deployment, t_3 be the time of deployment plus two hours, t_4 be the time of the start of data acquisition, t_5 be the time of the end of data acquisition, t_6 be the time of the post-recovery clock calibration (see Fig. 1). Precise clock calibrations are available at t_1, t_2 and t_6.
(2) Extend the pre-deployment clock drift rate, determined from the clock calibrations at t_1 and t_2, for two hours beyond the time of deployment to estimate the clock correction at time t_3. This is to allow for the thermal inertia of the instrument package after deployment.
(3) Assuming that the clock drifts at the cold-temperature rate between time t_3 and time t_4 and again between time t_5 and time t_6, and at the slightly-warmer-temperature rate between time t_4 and time t_5, and using the

estimated drift-rate difference between the cold and slightly-warmer temperatures, compute the clock correction at the time of the data acquisition and the clock drift rate during the data-acquisition-phase (i.e., the slightly-warmer-temperature rate).

(4) If more than one data acquisition modes (in terms of number of channels and sampling intervals) are used, adjust the estimation procedure accordingly for different estimated temperature rises at different phases.

The above estimation can be conveniently programmed on an **Excel** spreadsheet on a Macintosh. It can also be programmed on a SUN, or even a hand calculation is workable.

More precise estimates of clock corrections will be made later using the in-situ clock calibration obtained simultaneously with the determination of OBS locations.

Sampling Rate Correction

Each OBS CPU board carries two crystal-controlled oscillators: one for the real-time clock and the other for the baud-rate generator for serial communication. The latter is also used to control the sampling rate of the data acquisition. Since these two are independent oscillators not synchronized to each other, the sampling time drifts relative to the real time. This relative drift may amount to a few tens of milliseconds at the end of each record, enough to show up as visible steps in arrivals on a seismic record section. The baud-rate (sampling-rate) generator is calibrated against the real-time clock by measuring the duration of time from the end of data acquisition for a record to the next real-time clock update, which occurs every 100 ms. This time duration is measured in units of 77/(9600×16) s, or 0.5013 ms, and the time count, called the residual count, is stored in the seventh byte of the header of the last block of each record (see Appendix A).

Compare the recorded residual count, plus 1, 2, 4, or 5 for 1-, 2-, 3-, or 4-channel data acquisition, respectively, to compensate for the amount of time spent on software, with what is expected if the crystals were exactly matched. The difference in the residual count multiplied by the time unit of 0.5013 ms and divided by the number of samples per record per channel will give the deviation of the sampling interval.

This computation can be done conveniently on an **Excel** spreadsheet on a Macintosh, but it can also be programmed on any computer, or can be performed by a hand calculation.

Shot Location

Each shot must be located in the coordinate system used for the project. This is done by using the navigation data, the times of shots, and the knowledge of the source setback, the distance between the location of the antenna used to receive the navigation signal and the center of the source array. The raw navigation data often contain many irregularities and individual location fix is usually associated with appreciable errors. Irregularities need be removed, and errors need be reduced. How this can be done depends on the type of navigation data. Smoothing of navigation data to average out the random errors is often necessary. The objective here is to obtain a satisfactory estimate of each shot location, which is derived by estimating the location of the navigation receiving antenna at the time of the shot and displacing it by the amount of the source setback in the direction opposite to the ship's heading. There are several

programs used in the past for this purpose. For example, program **lrnfit** computes piece-wise-continuous smoothing functions for Loran-C TD's, to be used for estimating shot locations, and program **shotll** estimates shot locations by using linear regression of raw latitude-longitude data, such as those from a GPS receiver, within a specified window length.

The quality of the shot locations thus estimated should be checked to see if they are reasonable. One way to do this is to compute the ship's speed and heading from the computed shot locations, and to plot them on a computer screen to detect any unreasonably large speed or course changes. If the results show unrealistic movement of the ship, then we need to find the cause of the problem, correct them and recompute the shot locations. Note that certain systematic errors, such as those intentionally introduced into the GPS data by the 'selective availability,' may not be detectable at this stage, but may show up later as irregular water-wave arrivals.

Bathymetry Correction

The raw bathymetry data usually assume a constant sound speed through the entire column of water between the transducer and the sea floor in computing the water depth from the two-way transit time of sound echo. Since the speed of sound in sea water depends on salinity and temperature, a correction is required to obtain correct water depths from the bathymetry data. In absence of actual temperature and salinity measurements, such a correction can be estimated for a given region using past empirical data. We normally use the so-called Matthews' table compiled and published by the British Hydrographic Office. The latest edition is by *Carter* [1980].

OBS Location and Orientation

The precise location of each OBS on the sea floor is determined using the direct water-wave arrivals recorded by the OBS. When horizontal-component data are recorded, such data can also be used in OBS location as well as orientation of the instrument on the sea floor. We follow the procedure described by *Nakamura et al.* [1987].

The first step is to select the sections of data from shots fired near each OBS in which direct water-wave arrivals are clearly recorded. Generally, those shots fired within the range equal to the water depth, thus giving water-wave incident angles of less than about 45°, provide good water-wave data for OBS location. Program **disest** may be used to obtain rough estimates of distances from shots to OBS's. It is often more convenient to make a separate small file which is a subset of the original OBS data file containing only the data needed for locating each OBS. Program **CopySelBlocks** or its variation **CopySelBlocks3** may be used for this purpose. Also, the knowledge of the record/block numbers in which the actual water-wave onsets are recorded may be helpful in picking the arrivals. This information may be estimated from the shot times, OBS record header times and clock corrections, using a program such as **FindShotBlocks**, or picked more precisely from a compressed-scale display of the data using a programs such as **OBSPlot.CMP/c3** on a Macintosh.

The next step is to pick the water-wave arrivals and measure the arrival times and, if horizontal-component data are recorded, the ratio of the two horizontal amplitudes. The water-wave arrivals can be picked from a preliminary record-section plot by hand, but it is generally more expedient to do it automatically on a computer. To do this, however, requires a use of an event-detection algorithm appropriate for the given

characteristics of the arrivals. It may be possible to write a general-purpose event-detection program, but it is often more satisfactory to use a specific detection program especially written for the recorded water-wave arrivals, taking into account such factors as the nature of the background noise, source signature, filter response, and sampling interval. For each of the picked arrival, compute the arrival time. This is done by taking the record header time, and adding the displacement based on the block and sample numbers where the arrival is picked and the sampling interval as corrected above. The record header time refers to the time one sample interval before the time of the first sample of the record. There are several different versions of programs used in the past to do the arrival picks. They include **ArrPick3c**, **ArrPick1c**, and several others. They may be examined to see if any of them can be used for the particular data set being processed.

The next step is to compute the travel times by subtracting the shot times from the arrival times after correcting the latter for drift. Program **TTComp** can be used for this purpose.

The final step of OBS location is a least squares inversion of a set of travel-time data, and the polarization angles of the horizontal component particle motions if recorded, to obtain a set of unknown parameters that include at least the two coordinates of the OBS location and an additional clock correction, and may include others such as orientation of the horizontal components and average water-wave speed. What parameters can be reliably determined depends upon the set of available data in relation to the relative geometry of the shot points near the OBS. Programs used in the past on Macintosh include **OBSLoc3** to compute geographic coordinates and clock correction from arrival times only, **OBSLoc4** to compute geographical coordinates, OBS orientation and clock correction, and **OBSLoc5xy** to compute UTM coordinates, OBS orientation, clock correction and average water-wave speed, to give just a few examples.

It is often useful to plot the measured and theoretical (based on the computed locations and orientation) travel times and horizontal polarization on a computer screen. This serves three purposes: (1) any problems in picking the arrivals should be apparent; (2) problems in navigation data may show up as irregular water-wave arrival times; and (3) the goodness of the fit can be ascertained. Program **WTAPlot** can be used on Macintosh for this purpose (Fig. 2).

Final Clock Correction

The preliminary clock drift correction estimated earlier should be further refined using the clock correction at the time of ship's passage over the OBS obtained with the OBS location above. This, again can be conveniently programmed on an **Excel** spreadsheet on a Macintosh.

SEG-Y Tape Generation

First, gather all the data and parameters needed for the SEG-Y tape generation. The need data files are:

(1) a file containing shot number, shot time, shot coordinates and water depth for each shot (**ShotNTLLD.ext**);
(2) a file containing OBS header information (**HdrList.ext**); and
(3) a file containing the OBS data (**obsdata.ext**).

Other data and parameters needed are:

(1) number of output channels desired;
(2) precise sampling interval, as corrected for the drift rates of the baud-rate generator and the real-time clock;
(3) desired time window for each trace expressed in terms of advance window opening, window shift rate with offset, and minimum trace length;
(4) corrected OBS depth;
(5) source (air-gun) depth;
(6) OBS location coordinates;
(7) anti-aliasing filter frequency;
(8) clock correction and drift rate;
(9) shot delay, if any; and
(10) OBS orientation, if applicable.

Additional parameters may be needed in special cases.

Program **SegySetup**, currently written for a Macintosh, uses all of the above data and parameters except the OBS data and OBS orientation and generates a file named **SegyPar.ext** that contains the information needed to generate the trace header and to find the OBS data for each trace. The parameters may be inputted from the keyboard, or from a text file named **SegySetupPar.ext**. The keyboard-inputted parameters may be saved for later revisions and reuse.

Program **segy3**, (**segy.f** compiled and linked with **convert3.f** and **intibm.f**) uses the information contained in the file **SegyPar.ext** to generate SEG-Y trace headers, finds the data in **obsdata.ext** file, converts the OBS-format data to IBM floating point format, and generates a SEG-Y format binary header file named **BinHdr.ext** and a SEG-Y format data file named **SegyData.ext**. Optionally, the program also accepts the OBS orientation from the keyboard and generates a SEG-Y data file with rotated horizontal-component data in channels 2 (radial) and 3 (transverse).

SEG-Y tape headers can be written first in ASCII using any text editor. Program **SegYAtoE** or **SegYAtoE/C**, to be run on a Macintosh, then converts the ASCII files to EBCDIC files required for SEG-Y format tapes. Program **SegYAtoE** requires a separate input ASCII file for each header to be generated, while program **SegYAtoE/C** accepts a single input file containing multiple ASCII headers, and thus is more efficient when there are several sets of OBS data to be processed. The output EBCDIC files are named **EbcHdr.ext**.

Program **segytape** combines the two tape header files, **EbcHdr.ext** and **BinHdr.ext**, and the data file **SegyData.ext** and writes a SEG-Y format 9-track tape. A similar program **segytapex** writes the output to an Exabyte tape.

Program **segyqc** reads a SEG-Y format tape, either 9-track or Exabyte, and generates a QC output.

Program **readsegy** may be used to retrieve the SEG-Y format files from either a 9-track or an Exabyte tape to disk files **EbcHdr.ext**, **BinHdr.ext** and **SegyData.ext**.

REFERENCES

Carter, D. J. T., Echo-Sounding Correction Tables (formerly Matthew' Tables), third ed., Hydrographic Department, Ministry of Defense, Taunton, 1980.

Nakamura, Y. and J. Garmany, Development of upgraded ocean-bottom seismograph, Technical Report No. 111, University of Texas Institute for Geophysics, Austin, 1991.

Nakamura, Y., P. L. Donoho, P. H. Roper, and P. M. McPherson, Large-offset seismic surveying using ocean-bottom seismographs and air guns: Instrumentation and field technique, *Geophysics, 52*, 1601-1611, 1987.

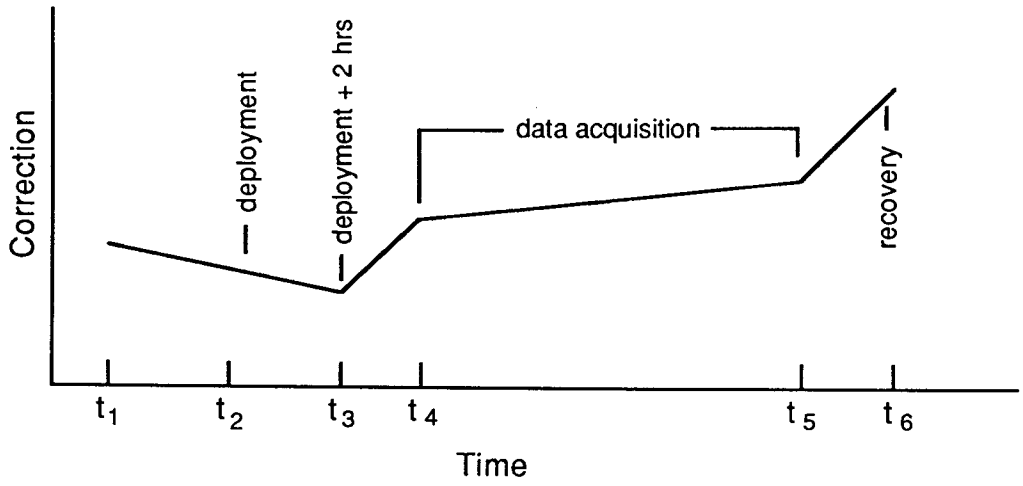

Fig. 1. Schematic representation of clock drift

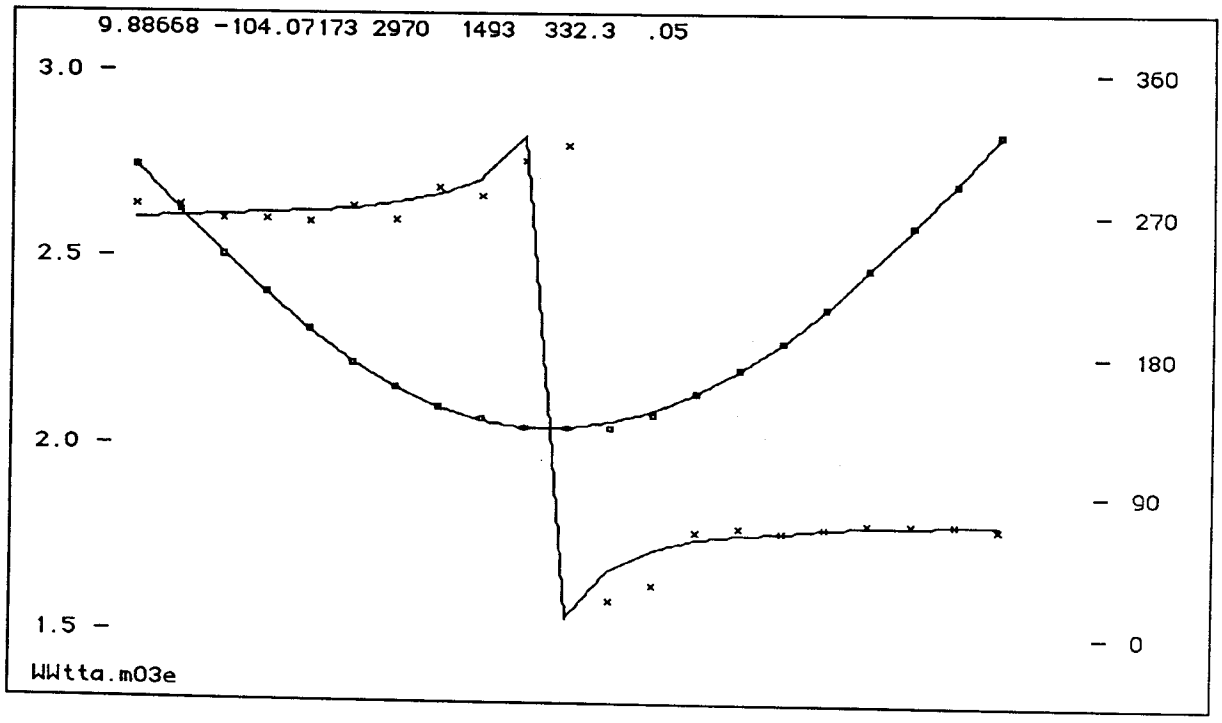

Fig. 2. An example of water-wave arrival times (squares with left scale, sec) and horizontal polarization (crosses with right scale, degrees).

Appendix A. Header Format

Header occupies the first 16 bytes of each 4K block. All header entries are in binary.

Byte	Data
0	Record number, low byte
1	Record number, high byte
2	Block number
3	Number of blocks in this record
4	Number of channels in low nibble (high nibble reserved)
5	Sampling interval in ms
6	Sampling residual count* (last block of each record only)
7	Reserved for earthquake OBS
8	OBS station ID number
9	Year
10	Month
11	Day
12	Hour
13	Minute
14	Second
15	1/10 second

*The time interval between the end of the data acquisition for each record and the next clock update, in units of 0.5013 ms. The count is to be used for calibrating the sampling timer against the real-time clock. The deviation, δ in ms, of the actual sampling interval from the theoretical interval, τ_t in ms, is given by

$$\delta = \{100[1 - \text{frac}(n\,\tau_t/100)] - 0.5013\,(r + c)\}/n$$

where r is the residual count, n is the number of samples per record per channel, and c (= 1, 2, 4, and 5 for 1-, 2-, 3-, and 4-channel acquisition, respectively) is the correction due to the additional time spent in software outside the counting period.

Appendix B. Data Format

Data occupies the last 4080 bytes of each 4K block. Two bytes (16 bits) per data sample, multiplexed (i.e., channel 1, ch. 2, ..., ch. 1, ch. 2, ...). Channels 1, 2, 3 and 4 are assigned to vertical, horizontal 1, horizontal 2 and hydrophone, respectively. In each data sample:

Byte	Bits	Data
1st	6 msb	6 lsb of ADC output
1st	2 lsb	Exponent of power 5
2nd	all	8 msb of ADC output

ADC output is in offset binary with 2000h for zero input. To obtain amplitude referred to the highest gain, subtract 2000h from ADC and multiply by 5 raised to the power of the exponent.

Appendix C. Actual Sampling Intervals

Nominal Sampling Interval ms	Actual Sampling Interval ms	Error
1	1.0026	0.260%
2	1.9987	-0.065%
3	3.0013	0.043%
4	3.9974	-0.065%
5	5.0000	0.000%
6	6.0026	0.043%
7	6.9987	-0.019%
8	8.0013	0.016%
9	8.9974	-0.029%
10	10.0000	0.000%
11	11.0026	0.024%
12	11.9987	-0.011%
13	13.0013	0.010%
14	13.9974	-0.019%
15	15.0000	0.000%
16	16.0026	0.016%
17	16.9987	-0.008%
18	18.0013	0.007%
19	18.9974	-0.014%
20	20.0000	0.000%
21	21.0026	0.012%
22	21.9987	-0.006%
23	23.0013	0.006%
24	23.9974	-0.011%
25	25.0000	0.000%

Appendix D. Processing Programs — SUN FORTRAN

Program listings of the following sample SUN FORTRAN programs are given in the following pages:

ArrPick1c.f	arrival picking for single-channel, continuously recorded data
ArrPick3c.f	arrival picking for 3-channel, contiguously recorded data
AssShsotNos.f	assigning shot numbers to shots
convert3.f	decoding of raw OBS data
CopySelBlocks.f	copying of selected raw OBS data blocks
datacorrc44.f	byte shift error correction of raw OBS data
disest.f	rough estimation of distances between OBS and shots
FindShotBlocks.f	estimation of OBS data block/record numbers for given shot times
HdrTimInt.f	decimal interpretation of hexadecimal header listing and time intervals
HeaderListA.f	hexadecimal header listing of all blocks of raw OBS data
HeaderListL.f	hexadecimal header listing of the last block of each OBS raw data record
intibm.f	format conversion from integer to IBM-floating
lrnfit.f	smoothing of Loran-C TD values
readsygy.f	transferring SEG-Y format data from a 9-track or Exabyte tape to SUN disk files.
segy.f	generation of SEG-Y format files
segyqc.f	quality control of SEG-Y format tapes
segytape.f	writing of SEG-Y format data to a 9-track tape
segytapex.f	writing of SEG-Y format data to an Exabyte tape
shotll.f	smoothing of latitude-longitude values
TTComp.f	travel time computation

ArrPick1c.f

```
      program ArrPick1c
c to pick water-wave arrivals for time only
c continuous input data **20-Jul-92**
      byte in(4096),hdr(8)
      integer*2 data(2040)
      character ext*8
      equivalence (in(17),data(1))
      common a(2040),in,n,n2,nb,nch,nsc,ip,jf
c
      write(*,"('Open obsdata.'$)")
      read(*,'(a)') ext
      open(1,file='obsdata.'//ext,access='direct',form='unformatted'
&     ,recl=4096)
      read(1,rec=1) in
      nch=in(5)
      nsc=2040/nch
      si=in(6)
      si=int(153.6*si+.5)/153600.
      sims=si*1000.
      print *, nch,' channels  ',nsc,' samples  si =',sims,' ms'
      write(*,"('Correction to si, in ns: '$)")
      read(*,*) sic
      si=si+sic*1.e-9
      ips=10./si
      write(*,"('Starting and ending record Nos.: '$)")
      read(*,*) n1,n2
      write(*,"('Thresholds, relative and absolute: '$)")
      read(*,*) thr,tha
      open(2,file='ArrPicks.'//ext)
      n=n1-1
c
      call getablock
10    hdr(1)=in(2)
      hdr(2)=in(1)
      do i=3,8
       hdr(i)=in(i+8)
       end do
      write(*,"(2z2.2,1x6z2.2,i4)") hdr,nb
15    anois=0.
       zero=a(ip)
      do j=2,32
       ip=ip+1
16     if (ip.le.nsc) then
        anois=(anois*15.+abs(a(ip)-zero))/16.
        do k=1,3
         zero=(zero*63.+a(ip))/64.
         end do
       else
        call getablock
        if (jf) 16,18,90
18      continue
        end if
       end do
c
20    ip=ip+1
21    do while (ip.le.nsc)
       asig=abs(a(ip)-zero)
       if (asig.gt.anois*thr.and.asig.gt.tha) go to 30
       anois=(anois*63.+asig)/64.
        zero=(zero*63.+a(ip))/64.
       ip=ip+1
       end do
      call getablock
      if (jf) 10,21,90
c
30    if (ip.gt.1) then
       a0=a(ip-1)-zero
      else
       a0=as-zero
       end if
      a1=a(ip)-zero
      if (ip.eq.nsc) then
       call getablock
       if (jf) 10,32,90
32     continue
       end if
      a2=a(ip+1)-zero
      if ((abs(a0).gt.anois*3.and.a0*a1.gt.0.)
&     .or.abs(a1).ge.abs(a2)/2.) then
```

```
         ip=ip-1
         a2=a1
         a1=a0
         end if
        tt=a1/(a1-a2)
        tt=ip+sign(1.-(1.-abs(tt))**2,tt)
        t=(nsc*(nb-1)+tt)*si
        k1=a1
        k2=a2
        write(*,"(f6.1,2i6,i4,f7.1,f9.3)") anois,k1,k2,nb,tt,t
        write(2,"(2z2.2,1x6z2.2,2i6,i4,f7.1,f9.3)") hdr,k1,k2,nb,tt,t
        jp=ip+ips
        do while (jp.gt.nsc)
         call getablock
         if (jf) 10,45,90
45       jp=jp-nsc
         end do
        go to 15
c
90      close(1)
        close(2)
        stop
        end
c
        subroutine getablock
        byte in(4096)
        integer obsint
        integer*2 data(2040)
        equivalence (in(17),data(1))
        common a(2040),in,n,n2,nb,nch,nsc,ip,jf
c
        n=n+1
        if (n.le.n2) then
         lb=nb
         read(1,rec=n) in
         nb=in(3)
         if (nb.lt.0) nb=nb+256
         do i=1,nsc
          a(i)=obsint(data(nch*(i-1)+1))/8.
          end do
         ip=1
         if (nb.eq.lb+1) then
          jf=0
         else
          jf=-1
          end if
        else
         jf=1
         end if
        return
        end
```

ArrPick3c.f

```
        program ArrPick3c
c to pick water-wave arrivals for time and polarization
c continuous input data **20-Jul-92**
        byte in(4096),hdr(8)
        integer*2 data(2040)
        integer na(3,10)
        dimension zero(3),sa(2),sum(2),ssum(2),aa(2),as(3)
        character ext*8
        equivalence (in(17),data(1))
        common a(3,680),in,n,n2,nb,nch,nsc,ip,jf
c
        write(*,"('Open obsdata.'$)")
        read(*,'(a)') ext
        open(1,file='obsdata.'//ext,access='direct',form='unformatted'
&       ,recl=4096)
        read(1,rec=1) in
        nch=in(5)
        nsc=2040/nch
        si=in(6)
        si=int(153.6*si+.5)/153600.
        sims=si*1000.
        print *, nch,' channels  ',nsc,' samples  si =',sims,' ms'
        write(*,"('Correction to si, in ns: '$)")
        read(*,*) sic
        si=si+sic*1.e-9
        ips=10./si
        write(*,"('Starting and ending record Nos.: '$)")
```

```
      read(*,*) n1,n2
      write(*,"('Thresholds, relative and absolute: '$)")
      read(*,*) thr,tha
      open(2,file='ArrPicks.'//ext)
      n=n1-1
c
      call getablock
10    hdr(1)=in(2)
      hdr(2)=in(1)
      do i=3,8
       hdr(i)=in(i+8)
       end do
      write(*,"(2z2.2,1x6z2.2,i4)") hdr,nb
15    anois=0.
      do k=1,3
       zero(k)=a(k,ip)
       end do
      do j=2,32
       ip=ip+1
16     if (ip.le.nsc) then
        anois=(anois*15.+abs(a(1,ip)-zero(1)))/16.
        do k=1,3
         zero(k)=(zero(k)*63.+a(k,ip))/64.
         end do
       else
        call getablock
        if (jf) 16,18,90
18      continue
        end if
       end do
c
20    ip=ip+1
21    do while (ip.le.nsc)
       asig=abs(a(1,ip)-zero(1))
       if (asig.gt.anois*thr.and.asig.gt.tha) go to 30
       anois=(anois*63.+asig)/64.
       do k=1,3
        zero(k)=(zero(k)*63.+a(k,ip))/64.
        end do
       ip=ip+1
       end do
      call getablock
      if (jf) 10,21,90
c
30    if (ip.gt.1) then
       a0=a(1,ip-1)-zero(1)
      else
       a0=as(1)-zero(1)
       end if
      a1=a(1,ip)-zero(1)
      if (ip.eq.nsc) then
       call getablock
       if (jf) 10,32,90
32     continue
       end if
      a2=a(1,ip+1)-zero(1)
      if ((abs(a0).gt.anois*3.and.a0*a1.gt.0.)
&     .or.abs(a1).ge.abs(a2)/2.) then
       ip=ip-1
       a2=a1
       a1=a0
       end if
      tt=a1/(a1-a2)
      tt=ip+sign(1.-(1.-abs(tt))**2,tt)
      t=(nsc*(nb-1)+tt)*si
      k1=a1
      k2=a2
      write(*,"(f6.1,2i6,i4,f7.1,f9.3)") anois,k1,k2,nb,tt,t
      if (k1*k2.lt.0) ip=ip+1
      do k=1,2
       sum(k)=0.
       ssum(k)=0.
       end do
      nbs=nb
      do j=1,10
35     if (ip.le.nsc) then
        do k=2,3
         sa(k-1)=a(k,ip)-zero(k)
         end do
        if (j.gt.3) then
         if (abs(sum(1)).gt.abs(sum(2))) then
```

```
            if (sa(1)*sum(1).lt.0.) go to 40
           else
            if (sa(2)*sum(2).lt.0.) go to 40
            end if
           end if
          na(1,j)=a(1,ip)-zero(1)
          do k=1,2
           na(k+1,j)=sa(k)
           sum(k)=sum(k)+sa(k)
           ssum(k)=ssum(k)+sa(k)**2
           end do
          ip=ip+1
         else
          call getablock
          if (jf) 10,35,90
          end if
         end do
c
40       jj=j-1
         do k=1,3
          write(*,"(10i6)") (na(k,j),j=1,jj)
          end do
         do k=1,2
          aa(k)=sign(sqrt(ssum(k)),sum(k))
          end do
         az=atan2(aa(2),aa(1))*57.29578
         if (az.lt.0.) az=az+360.
         print *, aa,az
         write(2,"(2z2.2,1x6z2.2,2i6,i4,f7.1,f9.3,f7.2)")
&         hdr,k1,k2,nbs,tt,t,az
         jp=ip+ips
         do while (jp.gt.nsc)
          call getablock
          if (jf) 10,45,90
45        jp=jp-nsc
          end do
         go to 15
c
90       close(1)
         close(2)
         stop
         end
c
         subroutine getablock
         byte in(4096)
         integer obsint
         integer*2 data(2040)
         equivalence (in(17),data(1))
         common a(3,680),in,n,n2,nb,nch,nsc,ip,jf
c
         n=n+1
         if (n.le.n2) then
          lb=nb
          read(1,rec=n) in
          nb=in(3)
          if (nb.lt.0) nb=nb+256
          do i=1,nsc
           do k=1,3
            a(k,i)=obsint(data(nch*(i-1)+k))/8.
            end do
           end do
          ip=1
          if (nb.eq.lb+1) then
           jf=0
          else
           jf=-1
           end if
         else
          jf=1
          end if
         return
         end
```

AssShotNos.f

```
         program AssShotNos
c assigned shot numbers for given set of shot times **12-Jun-92**
         character ext*8
c
         write(*,"('Open ShotTimes.'$)")
         read(*,'(a)') ext
```

```
        open(1,file='ShotTimes.'//ext)
        write(*,"('Assign shor number: '$)")
        read(*,*) is
        write(*,"('at (d,h,m,s): '$)")
        read(*,*) id,ih,im,sec
        write(*,"('Shot interval (s): '$)")
        read(*,*) si
        tb=(ih*60+im)*60+sec-si/2.
        open(2,file='ShotNT.'//ext)
c
10      read(1,"(i3,2(1xi2),1xf6.3)",end=99) kd,kh,km,sec
        if (kd.gt.0) then
         ns=((((kd-id)*24+kh)*60+km)*60+sec-tb)/si+is
         print 1, ns,kd,kh,km,sec
         write(2,1) ns,kd,kh,km,sec
1        format(2i4,2i3,f7.3)
         end if
        go to 10
c
99      close(1)
        close(2)
        stop
        end
```

convert3.f

```
c conversion of QADC-3 format OBS data  **28-Jul-91**
c
c conversion to integer
        function obsint(in1)
        implicit integer (a-z)
        integer*2 in1,in2
        obsint=(and(in1,255)*64+and(rshift(in1,10),63)-8192)
&        *5**(and(rshift(in1,8),3))
        return
c
c converstion to IBM floating
        entry obsibm(in2)
        fr=(and(in2,255)*1024+and(rshift(in2,6),1008)-131072)
&        *5**(and(rshift(in2,8),3))
        if (fr.gt.0) then
         ch=1157627904
        else if (fr.lt.0) then
         fr=-fr
         ch=-989855744
        else
         obsibm=1073741824
         return
         end if
        do while (fr.lt.1048576)
         fr=fr*16
         ch=ch-16777216
         end do
        obsibm=or(ch,fr)
        return
        end
```

CopySelBlocks.f

```
        program CopySelBlocks
c Copy selected OBS blocks to another file **05-Jun-92**
        byte data(4096)
        character ext*8
c
        write(*,"('Open input obsdata.'$)")
        read(*,'(a)') ext
        open(1,file='obsdata.'//ext,access='direct',form='unformatted'
&       ,recl=4096)
        write(*,"('Open WWBlocks.'$)")
        read(*,'(a)') ext
        open(2,file='WWBlocks.'//ext)
        open(3,file='obsdata.s'//ext,access='direct',form='unformatted'
&       ,recl=4096)
        mr=1
        nr=1
c
10      read(2,"(17xi3,f7.3)",end=99) jr,bl
        jb1=int(bl-0.5)
        jb2=jb1+1
        if (jb1.eq.0) jb1=1
        if (jb2.eq.129) jb2=128
c
20      read(1,rec=mr,end=99) data
```

```
        mr=mr+1
        kr=data(1)
        if (kr.lt.0) kr=kr+256
        kr=data(2)*256+kr
        if (kr-jr) 20,21,10
21      kb=data(3)
        if (kb.lt.0) kb=kb+256
        if (kb.lt.jb1) go to 20
        if (kb.eq.jb1.or.kb.eq.jb2) then
         write(3,rec=nr) data
         nr=nr+1
        end if
        if (kb.lt.jb2) go to 20
        go to 10
c
99      close(1)
        close(2)
        close(3)
        stop
        end
```

datacorrc44.f

```
        program DataCorrC44
c Correct byte shift error of C-44 OBS  **18-Sep-91**rev. 1-May-92**
        byte in1(4096),in2(4096),out1(4096),out2(4096)
        character fname*16,ext*8
        logical ib
        equivalence (in1(2),out1(1)),(out1(4096),in2(1)),(in2(2),out2(1))
c
        write(*,"('Open obsdata.'$)")
        read(*,'(a)') ext
        fname='obsdata.'//ext
        open(1,file=fname,access='direct',form='unformatted',recl=4096)
c
        read(1,rec=1) in1
        read(1,rec=2) in2
        if (in1(4)+1.eq.in2(4)) then
         write(1,rec=1) out1
         nr=3
         ib=.true.
c
10       if (ib) then
          read(1,rec=nr,end=91) in1
          if (in1(4).eq.1) then
           out2(4096)=out2(4096-2*out2(5))
           nro=out2(1)
           if (nro.lt.0) nro=nro+256
           nro=out2(2)*256+nro
           print *, nro
          else
           out2(4096)=in1(1)
          end if
          write(1,rec=nr-1) out2
          ib=.false.
         else
          read(1,rec=nr,end=92) in2
          if (in2(4).eq.1) then
           out1(4096)=out1(4096-2*mod(out1(5),16))
           nro=out1(1)
           if (nro.lt.0) nro=nro+256
           nro=out1(2)*256+nro
           print *, nro
          end if
          write(1,rec=nr-1) out1
          ib=.true.
         end if
         nr=nr+1
         go to 10
c
91       out2(4096)=out2(4096-2*mod(out2(5),16))
         write(1,rec=nr-1) out2
         nro=out2(1)
         if (nro.lt.0) nro=nro+256
         nro=out2(2)*256+nro
         print *, nro
         go to 99
92       out1(4096)=out1(4096-2*mod(out1(5),16))
         write(1,rec=nr-1) out1
         nro=out1(1)
         if (nro.lt.0) nro=nro+256
```

```
         nro=out1(2)*256+nro
         print *, nro
        else
         print *, "Correction not needed."
        end if
c
99      close(1)
        stop
        end
```

disest.f

```
        program disest
c **15-May-93**
        implicit double precision (l,s-t)
        character sta*2,ext*8
c
        write(*,"('Open stll.'$)")
        read(*,'(a)') ext
        open(1,file='stll.'//ext)
        write(*,"('Station: '$)")
        read(*,'(a)') sta
        open(2,file='aprdis.'//sta)
        write(*,"('lat. long. (deg,min): '$)")
        read(*,*) latd,latm,lond,lonm
        lato=latd+latm/60.d0
        lono=lond+lonm/60.d0
        c=cos(lato/57.29578)
        dil=1.
        dl=0.
c
10      read(1,*,end=99) t,lat,lon
        kd=t
        t=(t-kd)*24.d0
        kh=t
        t=(t-kh)*60.d0
        km=t
        ks=(t-km)*60.d0
        dis=sqrt((lat-lato)**2+(lon-lono)**2)*111.111
        di=dis-dl
        if (dil.le.0..and.di.ge.0.) write(*,1) kd,kh,km,ks,dis
        write(2,1) kd,kh,km,ks,dis
1       format(4i3,f7.2)
        dl=dis
        dil=di
        go to 10
c
99      close(1)
        close(2)
        stop
        end
```

FindShotBlocks.f

```
        program FindShotBlocks
c Estimate data blocks in which water-wave onsets are found
c if shots were directly above OBS  **03-June-92**
        integer hdr(16),idm(12)
        implicit double precision (s-t)
        character ext*8
        data idm/0,31,59,90,120,151,181,212,243,273,304,334/
c
        write(*,"('Open ShotTimes.'$)")
        read(*,'(a)') ext
        open(1,file='ShotTimes.'//ext)
        read(1,1) id,jh,jm,ss
1       format(i3,2(1xi2),1xf6.3)
        write(*,"('Open HdrList.'$)")
        read(*,'(a)') ext
        open(2,file='HdrList.'//ext)
        open(3,file='WWBlocks.'//ext)
        write(*,"('Renge of record numbers: '$)")
        read(*,*) nr1,nr2
        write(*,"('Clock correction, s, at hr, mn: '$)")
        read(*,*) cc,kh,km
        tr=(kh*60+km)*60
        write(*,"('Clock drift rate, s/day: '$)")
        read(*,*) cdr
        cdr=cdr/86400.
        write(*,"('Water depth, m: '$)")
        read(*,*) wd
        wt=wd/1500
```

```
      tt=(jh*60+jm)*60+wt
c
10    read(2,"(i5,8(z3.2,z2.2))",end=99) nb,hdr
      nr=hdr(2)*256+hdr(1)
      if (nr.lt.nr1) go to 10
      if (nr.gt.nr2) go to 99
      kd=idm(hdr(11))+hdr(12)-id
      if (hdr(10)/4*4.eq.hdr(10).and.hdr(11).gt.2) kd=kd+1
      ti=(hdr(13)*60+hdr(14))*60+hdr(15)+hdr(16)/10.
      td=ti-tr
      if (td.lt.-43200.) then
       td=td+86400.d0
      else if (td.gt.43200.) then
       td=td-86400.d0
       end if
      ti=ti+cc+td*cdr+kd*86400.d0
      bl=hdr(6)*2.040/hdr(5)
      do while (tt.lt.ti)
       read(1,1, end=99) jd,jh,jm,ss
       tt=(((jd-id)*24+jh)*60+jm)*60+ss+wt
       end do
15    bn=(tt-ti)/bl+1.
      if (bn.gt.129.) go to 10
      print 2, jd,jh,jm,ss,nr,bn
      write(3,2) jd,jh,jm,ss,nr,bn
2     format(3i3,f7.3,i4,f7.2)
      read(1,1, end=99) jd,jh,jm,ss
      if (jd.eq.0) go to 99
      tt=(((jd-id)*24+jh)*60+jm)*60+ss+wt
      go to 15
c
99    close(1)
      close(2)
      close(3)
      stop
      end
```

HdrTimInt.f

```
      program HdrTimInt
c OBS header time interval check **26-May-92**
      integer hdr(16)
      implicit double precision (s-t)
      logical first
      character ext*8
c
      write(*,"('Open HdrList.'$)")
      read(*,'(a)') ext
      open(1,file='HdrList.'//ext)
      open(2,file='HdrInt.'//ext)
      first=.true.
c
10    read(1,"(i5,8(z3.2,z2.2))",end=99) nb,hdr
      nr=hdr(2)*256+hdr(1)
      ny=1900+hdr(10)
      sec=hdr(15)+hdr(16)/10.d0
      tc=(hdr(13)*60+hdr(14))*60+sec
      if (first) then
       first=.false.
       print 1, hdr(9),ny
       write(2,1) hdr(9),ny
1      format('Station',i3,5x'Year',i5
&     /'block rec mo dy hr mn  sec   dt,s   tx,s')
       print 2, nb,nr,(hdr(i),i=11,14),sec
       write(2,2) nb,nr,(hdr(i),i=11,14),sec
2      format(i5,i4,4i3,f5.1,f7.1,f7.3)
      else
       td=tc-tl
       if (td.lt.0.d0) td=td+86400.d0
       tx=td-ta
       print 2, nb,nr,(hdr(i),i=11,14),sec,td,tx
       write(2,2) nb,nr,(hdr(i),i=11,14),sec,td,tx
      end if
      ta=int(hdr(6)*153.6+.5)/153600.d0*(2040/hdr(5)*hdr(4))
      tl=tc
      go to 10
c
99    close(1)
      close(2)
      stop
      end
```

HeaderListA.f

```
      program HeaderListA
c List OBS headers, all  **20-Aug-92**
      byte data(4096),header(16)
      character fname*16,ext*8
      equivalence (data(1),header(1))
c
      write(*,"('Open obsdata.'$)")
      read(*,'(a)') ext
      fname='obsdata.'//ext
      open(1,file=fname,access='direct',form='unformatted',recl=4096)
      open(2,file='HdrList.'//ext)
      lrn=0
      nr=1
c
10    read(1,rec=nr,end=99) data
       print 1, nr,header
       write(2,1) nr,header
1      format(i5,8(z3.2,z2.2))
      nr=nr+1
      go to 10
c
99    close(1)
      close(2)
      stop
      end
```

HeaderListL.f

```
      program HeaderListL
c List OBS headers  **25-May-92**
      byte data(4096),header(16)
      character fname*16,ext*8
      equivalence (data(1),header(1))
c
      write(*,"('Open obsdata.'$)")
      read(*,'(a)') ext
      fname='obsdata.'//ext
      open(1,file=fname,access='direct',form='unformatted',recl=4096)
      open(2,file='HdrList.'//ext)
      lrn=0
      nr=1
c
10    read(1,rec=nr,end=99) data
      if (header(3).eq.header(4)) then
       print 1, nr,header
       write(2,1) nr,header
1      format(i5,8(z3.2,z2.2))
      end if
      nr=nr+1
      go to 10
c
99    close(1)
      close(2)
      stop
      end
```

intibm.f

```
      function intibm(in)
c convert integer (<24 bits) to ibm floating  **28-Jul-91**
      implicit integer (a-z)
      fr=in
      if (fr.gt.0) then
       ch=1174405120
      else if (fr.lt.0) then
       fr=-fr
       ch=-973078528
      else
       intibm=1073741824
       return
       end if
      do while (fr.lt.1048576)
       fr=fr*16
       ch=ch-16777216
       end do
      intibm=or(ch,fr)
      return
      end
```

lrnfit.f

```
C       PROGRAM LRNFIT        LORAN-C CUBIC HERMITE FUNCTION FIT
C ********** 19-JUL-83 ***** REV. 16-DEC-83 **********
        character*7 INF,OUTF
        character*4 ext
        LOGICAL LAST
        implicit DOUBLE PRECISION (a-d,o-z)
        DIMENSION TD(2),TDB(2),Y(2),P(6),Q(6),R(5,2),S(5,2),G(4,4),H(4,2)
     1,E(2,2),RMSE(2),SD(2)
        DATA INF,OUTF/'loranc.','lorchf.'/,K/1/,LAST/.FALSE./
        NMF(I)=I-I/100*40+(ND-NDB)*1440
C
        print 1
1       FORMAT(' file loranc.'$)
        read 2, ext
2       FORMAT(a4)
        OPEN (UNIT=1,file=INF//ext)
        OPEN (UNIT=2,file=OUTF//ext)
        READ(1,*) ND,NHM,SEC,TD
        NDB=ND
        DO 15 J=1,2
15      TDB(J)=TD(J)-dMOD(TD(J),100.d0)
        print *, NDB,TDB
        WRITE(2,*) NDB,TDB
        WRITE(3,*) NDB,TDB
        print 4
4       format(' Base interval (min):'$)
        read *, ib
        bi=ib
C
20      NMB=NMF(NHM)/ib*ib
        M=0
        DO 25 I=1,5
        P(I)=0.
        DO 25 J=1,2
25      R(I,J)=0.
        P(6)=0.
C
30      X=(NMF(NHM)-NMB+SEC/60.)/bi
        IF (X.GT.1.) GO TO 40
        DO 33 J=1,2
        Y(J)=TD(J)-TDB(J)
33      R(5,J)=R(5,J)+Y(J)**2
        M=M+1
        XX=1.
        DO 37 I=1,6
        IF (I.GT.4) GO TO 36
        DO 35 J=1,2
35      R(I,J)=R(I,J)+XX*Y(J)
36      XX=XX*X
37      P(I)=P(I)+XX
        READ(1,*,END=39) ND,NHM,SEC,TD
        GO TO 30
39      LAST=.TRUE.
C
40      GO TO (41,42,42),K
41      K=2
        GO TO 70
42      N=N+M
        AM=M
        AN=N
        Q(1)=Q(1)+P(1)+AM
        Q(2)=Q(2)+P(2)+2.*P(1)+AM
        Q(3)=Q(3)+P(3)+3.*(P(2)+P(1))+AM
        Q(4)=Q(4)+P(4)+4.*(P(3)+P(1))+6.*P(2)+AM
        Q(5)=Q(5)+P(5)+5.*(P(4)+P(1))+10.*(P(3)+P(2))+AM
        Q(6)=Q(6)+P(6)+6.*(P(5)+P(1))+15.*(P(4)+P(2))+20.*P(3)+AM
        DO 43 J=1,2
        S(1,J)=S(1,J)+R(1,J)
        S(2,J)=S(2,J)+R(2,J)+R(1,J)
        S(3,J)=S(3,J)+R(3,J)+2.*R(2,J)+R(1,J)
        S(4,J)=S(4,J)+R(4,J)+3.*(R(3,J)+R(2,J))+R(1,J)
43      S(5,J)=S(5,J)+R(5,J)
        IF (K.EQ.3) GO TO 50
        DO 49 J=1,2
        DO 47 I=1,4
        DO 45 II=1,4
        L=8-I-II
        IF (L.NE.0) G(I,II)=Q(L)
45      CONTINUE
```

```
47      H(I,J)=S(5-I,J)
        G(4,4)=AN
49      CALL SIMQ(G,H(1,J),4,16,KS)
        K=3
        GO TO 60
C
50      DO 59 J=1,2
        DO 57 I=1,2
        DO 55 II=1,2
        L=8-I-II
55      E(I,II)=Q(L)
57      H(I,J)=S(5-I,J)-H(3,J)*Q(5-I)-H(4,J)*Q(4-I)
59      CALL SIMQ(E,H(1,J),2,4,KS)
C
60      DO 65 J=1,2
        A=H(1,J)
        B=H(2,J)
        C=H(3,J)
        D=H(4,J)
        SSD=(A*Q(6)+2.*B*Q(5))*A+(B*B+2.*A*C)*Q(4)+2.*(A*D+B*C)*Q(3)
     1 +(C*C+2.*B*D)*Q(2)+(2.*C*Q(1)+D*AN)*D+S(5,J)
        DO 63 I=1,4
63      SSD=SSD-2.*H(I,J)*S(5-I,J)
        SS=SSD
        RMSE(J)=dSQRT(SS/AN)
65      SD(J)=RMSE(J)/dSQRT(AN-2.)
        NHMX=NM0+NM0/60*40
        print 3, NHMX,H,RMSE,SD,N
        WRITE(3,3) NHMX,H,RMSE,SD,N
        WRITE(2,3) NHMX,H
        IF (LAST) GO TO 90
3       FORMAT(I5,3F8.3,2F9.3,2F8.3,2F9.3,3F8.3,I5)
        DO 69 J=1,2
        H(4,J)=H(1,J)+H(2,J)+H(3,J)+H(4,J)
69      H(3,J)=3.*H(1,J)+2.*H(2,J)+H(3,J)
C
70      N=M
        DO 73 I=1,5
        Q(I)=P(I)
        DO 73 J=1,2
73      S(I,J)=R(I,J)
        Q(6)=P(6)
        NM0=NMB
        if (x.gt.2.) k=1
        GO TO 20
C
90      STOP
        END
```

readsegy.f

```
        program readsegy
c Read a SEG-Y format tape
c  Output files: EbcHdr., BinHdr. and SegyData.
c ** 16-Sep-92 ** rev. 22-oct-92 **
        character drv*16,filename*16,ext*8,skip*8,size*8,cmd*64
        integer*2 binhdr(200),samples
        equivalence (binhdr(11),samples)
c
        write(*,"('/dev/n'$)")
        read(*,'(a)') drv
        drv='/dev/n'//drv(1:11)
c
10      write(*,"('Tape files to skip = '$)")
        read(*,'(a)') skip
        if (skip.ne.' ') call system('mt -f '//drv//' fsf '//skip)
        write(*,"('File name extension .'$)")
        read(*,'(a)', end=99) ext
        if (ext.eq.' ') go to 99
        cmd='dd if='//drv//' of=EbcHdr.'//ext//' bs=3200 count=1'
        print *, cmd
        call system(cmd)
        filename='BinHdr.'//ext
        cmd='dd if='//drv//' of='//filename//' bs=400 count=1'
        print *, cmd
        call system(cmd)
        open(1,file=filename,access='direct',form='unformatted',recl=400)
        read(1,rec=1) binhdr
        close (1)
        nb=samples*4+240
        write(size,*) nb
```

```
        i=1
        do while (size(i:i).eq.' ')
         i=i+1
         end do
        size=size(i:8)
        cmd='dd if='//drv//' of=SegyData.'//ext//' bs='//size
        print *, cmd
        call system(cmd)
        go to 10
c
99      stop
        end
```

segy.f

```
      program segy
c SEG-Y from c-44 OBS data ** 25-Jul-91** rev. 25-Sep-91 **
c %f77 segy.f convert[n].f intibm.f   [n=2 for QADC-2, n=3 for QADC-3]
        implicit integer (a-q,s-z)
        character filename*16,ext*8,exto*8
        integer*2 och,si,smpls,binhdr(200),in,th2(120),del
&       ,ff,yr,dy,hr,mn,sc,ms,smp,sd1,sd2,sd3
        integer trhdr(60),buf(8),cbl(16)
        byte ans,rot
        common in(2048,16),tsn,tsl,ffid,chan,shot,dd1(4),ofst,dd2(2),dag
&       ,dd3(2),dwt,dobs,dd4,slon,slat,olon,olat,dd5(5),del,sd1(15)
&       ,ff,sd2(7),yr,dy,hr,mn,sc,ms,dd7(3),az,smp,sd3(27)
&       ,out(8160),out2(8160)
        equivalence (binhdr(1),job),(binhdr(3),line),(binhdr(7),och)
&       ,(binhdr(9),si),(binhdr(11),smpls),(binhdr(31),ori)
&       ,(trhdr(1),tsn),(tsn,th2(1)),(cbl(1),sbl)
c clear header buffers
        do i=1,200
         binhdr(i)=0
         end do
        do i=1,60
         trhdr(i)=0
         end do
c input parameters
        write (*,"('Open SegyPar.'$)")
        read (*,'(a)') ext
        filename='SegyPar.'//ext
        print *, 'Opening ',filename
        open(1,file=filename)
        read (1,*) job,line,dobs,dag,olon,olat,ich,och,si,blks,smpls,ff,yr
        print *, 'job No.',job
        print *, 'line No.',line
        print *, 'OBS depth =',dobs,' m'
        print *, 'air-gun depth =',dag,' m'
        print *, 'OBS lat,long =',olat,olon,' ds'
        print *, 'i/o number of channels =',ich,' /',och
        print *, 'sampling interval =',si,' us'
        print *, 'number of blocks/trace =',blks
        print *, 'samples/trace =',smpls
        print *, 'alias filter cutoff =',ff,' Hz'
        print *, 'year =',yr
        tdbl=smpls*4+240
        print *, 'trace data block length =',tdbl,' bytes'
        spb=2040/ich
c optional horizontal rotation
        if (och.ge.3) then
         write (*,"('Rotate horizontals? '$)")
         read (*,'(a)') ans
         rot=ans.eq.'y'
         if (rot) then
          write (*,"('New job No. & H1 direction (deg. from N): '$)")
          read (*,*) job,rori
          ori=rori*100.
          end if
        else
         rot=.false.
         end if
c construct binary header
       binhdr(6)=1
       binhdr(10)=si
       binhdr(12)=-1
       do i=13,16
        binhdr(i)=1
        end do
       binhdr(28)=1
       print 1, (binhdr(i),i=1,32)
```

```
1       format(15z5)
        if (.not.rot) then
         filename='BinHdr.'//ext
        else
         filename='BinHdr.R'//ext
         end if
        print *, 'Opening ',filename
        open (3,file=filename,access='direct',form='unformatted',recl=400)
        write (3,rec=1) binhdr
        close (3)
        print *, 'Binary header saved in ',filename
c initialize trace header
        do i=15,18
         th2(i)=1
         end do
        trhdr(11)=-dobs
        th2(35)=1
        th2(36)=-10
        th2(45)=2
        th2(58)=smpls
        th2(59)=si
        th2(60)=3
        th2(72)=24
c open input and output data files
        write (*,"('Open obsdata.'$)")
        read (*,'(a)') exto
        filename='obsdata.'//exto
        print *, 'Opening ',filename
        open (2,file=filename,access='direct',form='unformatted'
&       ,recl=4096)
        if (.not.rot) then
         filename='SegyData.'//ext
        else
         filename='SegyData.R'//ext
         end if
        print *, 'Opening ',filename
        open(3,file=filename,access='direct',form='unformatted',recl=tdbl)
c construct trace data block
        irn=1
        tsn=1
50      read (1,*,end=99) trec,sbl,abl,ffid,shot,ofst,dwt,slon,slat,del
&       ,dy,hr,mn,sc,ms,az
        smp=abl*spb
        orec=0
        do while (orec.ne.trec.or.obl.ne.sbl)
         read (2,rec=irn,end=99) (in(i,1),i=1,2048)
         orec=and(in(1,1),255)*256+and(rshift(in(1,1),8),255)
         obl=and(rshift(in(2,1),8),255)
         if (tsn.le.6) print *, orec,obl
         if (orec.gt.trec.or.(orec.eq.trec.and.obl.gt.sbl)) then
          irn=irn-1
         else
          irn=irn+(trec-orec)*and(in(2,1),255)+sbl-obl
          end if
         end do
        irn=irn+1
        if (abl.gt.1) then
         do j=2,abl
          read (2,rec=irn,end=99) (in(i,j),i=1,2048)
          cbl(j)=and(rshift(in(2,j),8),255)
          irn=irn+1
          end do
         end if
        if (rot) then
         rb=(ori-az)/5729.578
         rs=sin(rb)
         rc=cos(rb)
         end if
        if (smp.lt.smpls) then
         do i=smp+1,smpls
          out(i)=0
          if (rot) out2(i)=0
          end do
         end if
        print *, tsn,orec," /",(cbl(j),j=1,abl)," /",smp
        do chan=1,och
         tsl=tsn
         if (tsn.le.6) then
          if (tsn.eq.1) print 1, (th2(i),i=1,93)
          print *, chan
          il=8+chan
```

```
          i2=7*ich+i1
          print 2, (in(i,1),i=i1,i2,ich)
2         format(8z9)
          j=1
          do i=i1,i2,ich
           buf(j)=obsint(in(i,1))
           j=j+1
          end do
          write (*,'(8i9)') buf
          end if
         k=1
         do blk=1,abl
          if (.not.rot.or.chan.eq.1.or.chan.eq.4) then
           do i=chan+8,2048,ich
            out(k)=obsibm(in(i,blk))
            k=k+1
           end do
          else if (chan.eq.2) then
           do i=10,2048,ich
            r1=obsint(in(i,blk))
            r2=obsint(in(i+1,blk))
            ar=-r1*rc+r2*rs
            at=-r1*rs-r2*rc
            out(k)=intibm(ar)
            out2(k)=intibm(at)
            k=k+1
           end do
          end if
         end do
        if (.not.rot.or.chan.ne.3) then
         write(3,rec=tsn) trhdr,(out(i),i=1,smpls)
        else
         write(3,rec=tsn) trhdr,(out2(i),i=1,smpls)
         end if
        if (tsn.le.6) then
         if (.not.rot.or.chan.ne.3) then
          print 2, (out(i),i=1,8)
         else
          print 2, (out2(i),i=1,8)
          end if
         if (tsn.eq.6) then
          write (*,"('Proceed? '$)")
          read(*,'(a)') ans
          if (ans.eq.'n') go to 99
          end if
         end if
        tsn=tsn+1
        end do
       go to 50
c
99     close (1)
       close (2)
       close (3)
       end
```

segyqc.f

```
c      program segyqc        SEG-Y tape quality check
c *** 17-Sep-92 ***
       implicit integer (a-z)
       character drv*16,rh1*3200,rh2*400,dbuf*30240,eh(40)*80
       integer*2 bh(200),th,smp
       logical ex
       common th(120),data(7500)
       equivalence (rh1,eh(1)),(rh2,bh(1)),(dbuf,th(1)),(smp,bh(11))
       data nf/0/,ex/.false./
c
       write(*,"('/dev/n'$)")
       read(*,'(a)') drv
       if (drv(1:1).eq.'v') ex=.true.
       drv='/dev/n'//drv(1:11)
       call system('mt -f '//drv//' rewind')
       open (2)
c
10     nb=topen(1,drv,.false.)
       if (nb.ne.0) call perror('open error')
       nb=tread(1,rh1)
       if (nb.eq.3200) then
         nf=nf+1
         print *, 'File',nf
         write(2,*) 'File',nf
```

```
          call e2a(rh1,3200)
          ll=40
          do while (eh(ll).eq.eh(ll-1))
            ll=ll-1
            end do
          do l=1,ll
            write(*,'(a)') eh(l)
            write(2,'(a)') eh(l)
            end do
        else if (nb.eq.0) then
          go to 99
        else
          print *, 'EBCDIC header illegible  nb=',nb
          end if
c
        nb=tread(1,rh2)
        if (nb.eq.400) then
          k=200
          do while (bh(k).eq.0)
            k=k-1
            end do
          write(*,'(10i7)') (bh(i),i=1,k)
          write(2,'(10i7)') (bh(i),i=1,k)
          lrec=smp*4+240
          trc=0
          ref=0
        else
          print *, 'binary header illegible  nb=',nb
          st=tclose(1)
          if (st.ne.0) call perror('close error')
          stop
          end if
c
        print 1
        write(2,1)
1       format('  rec#  tsq#    ffid    shot  year   day  hour   min'
&       ,'   sec    ms  #smp')
20      nb=tread(1,dbuf)
        if (nb.eq.lrec) then
          trc=trc+1
          if (th(8).eq.1) ref=ref+1
          ffid=th(6)
          if (ffid.lt.0) ffid=ffid+65536
          ffid=th(5)*65536+ffid
          shot=th(10)
          if (shot.lt.0) shot=shot+65536
          shot=th(9)*65536+shot
          if (th(2).ne.trc.or.ffid.ne.ref.or.
&          (th(93).ne.smp.and.th(8).eq.1)) then
            write(*,2) trc,th(2),ffid,shot,(th(i),i=79,84),th(93)
            write(2,2) trc,th(2),ffid,shot,(th(i),i=79,84),th(93)
2           format(2i6,2i8,7i6)
            ref=ffid
            end if
        else if (nb.eq.0) then
          write(*,2) trc,th(2),ffid,shot,(th(i),i=79,84),th(93)
          write(2,2) trc,th(2),ffid,shot,(th(i),i=79,84),th(93)
          write(2,*)
          st=tclose(1)
          if (st.ne.0) call perror('close error')
          if (ex) then
            call system('mt -f '//drv//' bsf 1')
            call system('mt -f '//drv//' fsf 1')
            end if
          go to 10
        else
          print *, 'tape read error  nb=',nb
          end if
        go to 20
c
99       t=tclose(1)
        if (st.ne.0) call perror('close error')
        call system('mt -f '//drv//' offline')
        stop
        end
```

segytape.f

```
        program segytape
c Write a 9-track SEG-Y format tapec
c  combining files EbcHdr., BinHdr., and SegyData.
```

```
c **29-Jul-91**
      character filename*16,ext*8,skip*8,size*8,cmd*64
      integer*2 binhdr(200),samples
      equivalence (binhdr(11),samples)
c
      write(*,"('Tape files to skip = '$)")
      read(*,'(a)') skip
      if (skip.ne.' ') call system('mt fsf '//skip)
c
10    write(*,"('File name extension .'$)")
      read(*,'(a)', end=99) ext
      if (ext.eq.' ') go to 99
      cmd='dd if=EbcHdr.'//ext//' of=/dev/nrmt8 bs=3200'
      print *, cmd
      call system(cmd)
      call system('mt bsf 1')
      filename='BinHdr.'//ext
      cmd='dd if='//filename//' of=/dev/nrmt8 bs=400'
      print *, cmd
      call system(cmd)
      call system('mt bsf 1')
      open(1,file=filename,access='direct',form='unformatted',recl=400)
      read(1,rec=1) binhdr
      close (1)
      nb=samples*4+240
      write(size,*) nb
      i=1
      do while (size(i:i).eq.' ')
       i=i+1
       end do
      size=size(i:8)
      cmd='dd if=SegyData.'//ext//' of=/dev/nrmt8 bs='//size
      print *, cmd
      call system(cmd)
      go to 10
c
99    call system('mt eof')
      stop
      end
```

segytapex.f

```
      program segytapex
c Write an Exbyte SEG-Y format tape
c  combining files EbcHdr., BinHdr., and SegyData.
c **16-Sep-92**
      character filename*16,ext*8,skip*8,size*8,cmd*64
      integer*2 binhdr(200),samples
      equivalence (binhdr(11),samples)
c
      write(*,"('Tape files to skip = '$)")
      read(*,'(a)') skip
      if (skip.ne.' ') call system('mt -f /dev/nvrsmt1 fsf '//skip)
c
10    write(*,"('File name extension .'$)")
      read(*,'(a)', end=99) ext
      if (ext.eq.' ') go to 99
      cmd='dd if=EbcHdr.'//ext//' of=/dev/nvrsmt1 bs=3200'
      print *, cmd
      call system(cmd)
      call system('mt -f /dev/nvrsmt1 bsf 1')
      filename='BinHdr.'//ext
      cmd='dd if='//filename//' of=/dev/nvrsmt1 bs=400'
      print *, cmd
      call system(cmd)
      call system('mt -f /dev/nvrsmt1 bsf 1')
      open(1,file=filename,access='direct',form='unformatted',recl=400)
      read(1,rec=1) binhdr
      close (1)
      nb=samples*4+240
      write(size,*) nb
      i=1
      do while (size(i:i).eq.' ')
       i=i+1
       end do
      size=size(i:8)
      cmd='dd if=SegyData.'//ext//' of=/dev/nvrsmt1 bs='//size
      print *, cmd
      call system(cmd)
      go to 10
c
```

```
99      call system('mt -f /dev/nvrsmt1 eof')
        stop
        end
```

shotll.f

```
        program shotll
c 12-May-93
        implicit double precision (l,s-t)
        character ext*8
        dimension tg(100),lag(100),log(100)
c
        write(*,"('Open ShotTimes and nav.'$)")
        read(*,'(a)') ext
        open(1,file='ShotTimes.'//ext)
        open(2,file='nav.'//ext)
        open(3,file='stll.'//ext)
        write(*,"('Time offset, hours: '$)")
        read(*,*) ito
        tofst=ito/24.d0
        write(*,"('Time window, +-sec: '$)")
        read(*,*) tw
        tw=tw/86400.d0
        m=1
        tg(1)=0.
c
10      read(1,"(i3,2(1xi2),1xf6.3)",end=99) kd,kh,km,ss
        ts=((ss/60.d0+km)/60.d0+kh)/24.d0+kd
        t1=ts-tw
        t2=ts+tw
        if (tg(m).ge.t1) then
         i=m-1
         do while (tg(i).ge.t1)
          i=i-1
          end do
         do j=i+1,m
          k=j-i
          tg(k)=tg(j)
          lag(k)=lag(j)
          log(k)=log(j)
          end do
         m=k
        else
         do while (tg(1).lt.t1)
          read(2,1) kd,kh,km,ks,klad,alad,klon,alon
1         format(i4,4(1xi2),f7.3,4xi3,f7.3)
          tg(1)=(((ks/60.d0)+km)/60.d0+kh)/24.d0+kd+tofst
          end do
         lag(1)=klad+alad/60.d0
         log(1)=klon+alon/60.d0
         m=1
         end if
        do while (tg(m).le.t2)
         m=m+1
         read(2,1,end=20) kd,kh,km,ks,klad,alad,klon,alon
         tg(m)=(((ks/60.d0)+km)/60.d0+kh)/24.d0+kd+tofst
         lag(m)=klad+alad/60.d0
         log(m)=klon+alon/60.d0
         end do
20      n=m-1
        if (n.gt.1) then
         st=0.d0
         sst=0.d0
         sla=0.d0
         stla=0.d0
         slo=0.d0
         stlo=0.d0
         do i=1,n
          td=tg(i)-ts
          st=td+st
          sst=td**2+sst
          sla=lag(i)+sla
          stla=td*lag(i)+stla
          slo=log(i)+slo
          stlo=td*log(i)+stlo
          end do
        tden=st**2-sst*n
        lat=(st*stla-sst*sla)/tden
        lon=(st*stlo-sst*slo)/tden
        write(*,2) ts,lat,lon
        write(3,2) ts,lat,lon
2       format(f12.8,2f10.5)
        end if
```

```
        go to 10
c
99      close(1)
        close(2)
        close(3)
        stop
        end
```

TTComp.f

```
        program TTComp
c compute travel time **12-Jun-92**
        implicit double precision (s-t)
        character ext*8
        integer idm(12)
        data idm/0,31,59,90,120,151,181,212,243,273,304,334/
c
        write(*,"('Open ShotNT.'$)")
        read(*,'(a)') ext
        open(1,file='ShotNT.'//ext)
        write(*,"('Open ArrPicks.'$)")
        read(*,'(a)') ext
        open(2,file='ArrPicks.'//ext)
        open(3,file='WWtta.'//ext)
        write(*,"('Clock correction (sec) = '$)")
        read(*,*) cc
        write(*,"('at (y,m,d,h,m) '$)")
        read(*,*) iy,im,id,ih,in
        t0=((id*24+ih)*60+in)*6.d1
        write(*,"('Clock drift rate (s/day) = '$)")
        read(*,*) cr
        cr=cr/86400.
        if (iy-iy/4*4.eq.0) then
         do i=3,12
          idm(i)=idm(i)+1
          end do
         end if
c
10      read(2,"(5x6z2.2,23xf9.3,f7.2)",end=99) km,kd,kh,kn,ks,kf,t,a
        t=((kd*24+kh)*60+kn)*60+ks+kf/1.d1+t
        dt=t-t0
        if (im.ne.km) dt=(idm(km)-idm(im))*86400.+dt
        t=dt*cr+cc+t
20      read(1,*,end=99) ns,jd,jh,jm,ss
        tt=t-((((jd-idm(km))*24+jh)*60+jm)*60+ss)
        if (tt.gt.15.) go to 20
        if (tt.gt.0.) then
         print 1, ns,tt,a
         write(3,1) ns,tt,a
1        format(i7,f7.3,f7.2)
         end if
        go to 10
c
99      close(1)
        close(2)
        close(3)
        stop
        end
```

Appendix E. Processing Programs — Macintosh QuickBasic

Program listings of the following sample Macintosh QuickBasic programs are given in the following pages:

ClockCalSelect	extracting clock calibration data from capture files
OBSPlot.CMP/c3	compressed scale plot of raw OBS data
OBSLoc3	OBS location and clock correction
OBSLoc4	OBS location, orientation and clock correction
SegYAtoE/C	ASCII to EBCDIC conversion of multiple SEG-Y headers
SegySetup	computing parameters needed for SEG-Y format tape generation
ShotTimeCheck	checking shot time logs for irregularities
WTAPlot	plotting of arrival water-wave times and polarization angles

ClockCalSelect

```
'Program ClockCalSelect  **26-May-92**
DEFINT a-z:TEXTFONT 4:TEXTSIZE 9
PRINT"Open input file.":fi$=FILES$(1,"TEXT"):IF fi$=""THEN END
GetID:INPUT"Station code: ",id$:IF id$=""THEN END
OPEN"I",1,fi$:OPEN"O",2,"ClockCals.s"+id$
WHILE NOT EOF(1):LINE INPUT#1,a$
IF MID$(a$,2,2)=id$ THEN PRINT a$:PRINT#2,a$
WEND:CLOSE 1:CLOSE 2:GOTO GetID
```

OBSPlot.CMP/c3

```
'Program OBSPlot.CMP/c3    OBS Plot:Compressed/C44/QADC-3
'  **13-May-91**
 DEFINT i-n:DIM STATIC ka(2039),kode(39)
 lf=0:jdk=0:shi=30!:dly=0!:ia=6:lim=3:ich=1:jvd=-1:jdd=-1:iy=4
 WINDOW 3,,(1,20)-(511,341),3:TEXTSIZE 9:TEXTFONT 4
 FOR i=0 TO 39:READ kode(i):NEXT
DATA &H4E56,&H0000,&H48E7,&H1020,&H206E,&H000E,&H226E,&H0008
DATA &H342E,&H000C,&H363C,&H07F8,&H1018,&H1200,&H1018,&HE140
DATA &H1001,&HE440,&H0240,&H3FFF,&H0440,&H2000,&H0281,&H0000
DATA &H0003,&H45FA,&H0018,&HD4C1,&H1212,&HC1C1,&HE4A0,&H32C0
DATA &H5343,&H66D4,&H4CDF,&H0408,&H4E5E,&H4E75,&H0105,&H197D
 MENU 8,0,0,"Command":MENU 8,1,1,"Open Disk File":MENU 8,2,0,"Plot"
 MENU 8,3,1,"  Change Attenuation":MENU 8,4,1,"  Change Clipping"
 MENU 8,5,0,"Abort":MENU 8,6,1,"Quit"
 GOSUB W2:ON MENU GOSUB Cmd:MENU ON
Wloop:WHILE NOT lf:WEND:END
W2:WINDOW 2,,(260,50)-(500,100),2
W2A:MENU 8,0,1:CLS:GOSUB PCenter:PRINT"Please select a command":RETURN
Cmd:IF MENU(0)<>8 THEN RETURN ELSE mn1=MENU(1):iab=0
 ON mn1 GOTO Disk,Plot,Atten,Clip,Abort,Quit
PCenter:GOSUB OpW2:LOCATE 2,2:RETURN
OpW2:IF WINDOW(1)=2 THEN RETURN ELSE WINDOW 2:RETURN
Disk:f$=FILES$(1,"TEXT"):IF f$=""GOTO W2A
 IF jdk THEN CLOSE 1
 OPEN"R",1,f$,4096:FIELD 1,4096 AS in$
 jdk=-1:MENU 8,2,1:GOTO W2A
Plot:CLS:GOSUB PCenter:PRINT f$:INPUT"  Starting disk record No. ",rcn
 WINDOW CLOSE 2:CLS:MENU 8,5,1:MENU ON:lr=0
 kx=0:ky=20:kb=0:nb=1:na=0
 WHILE kb<nb:IF iab GOTO Prtn
 GET 1,rcn:IF EOF(1) GOTO Prtn
 rcn=rcn+1:ih$=MID$(in$,1,16)
 IF kb=0 THEN
  h$="":FOR j=1 TO 16:ak=ASC(MID$(ih$,j,1))
  h$=h$+HEX$(ak\&H10)+HEX$(ak AND &HF):IF (j MOD 2)=0 THEN h$=h$+" "
  NEXT:PRINT h$;SPACE$(30);"Atten. =";ia;
  nb=ASC(MID$(ih$,4)):nch=ASC(MID$(ih$,5))MOD 16
  nch1=nch-1:ns=2040/nch:jy=288/nch
  FOR j=0 TO nch1:lmax(j)=0:lmin(j)=0:kmax(j)=-lim:kmin(j)=lim:NEXT
  END IF
 kb=ASC(MID$(ih$,3)):d$=MID$(in$,17)
 Decode=VARPTR(kode(0)):Decode SADD(d$),ia,VARPTR(ka(0))
 FOR i=0 TO ns-1:in=i*nch
  FOR j=0 TO nch1:ij=in+j:kij=ka(ij)
   IF kij>kmax(j)THEN kmax(j)=kij
   IF kij<kmin(j)THEN kmin(j)=kij
   NEXT:na=na+1
  IF na=8 THEN
   FOR j=0 TO nch1
    IF kmax(j)<lmin(j)THEN max=lmin(j)ELSE max=kmax(j)
    IF kmin(j)>lmax(j)THEN min=lmax(j)ELSE min=kmin(j)
    IF max>lim THEN max=lim ELSE IF max<-lim THEN max=-lim
    IF min>lim THEN min=lim ELSE IF min<-lim THEN min=-lim
    ly=ky+j*jy:LINE(kx,ly-max)-(kx,ly-min)
    lmax(j)=kmax(j):lmin(j)=kmin(j):kmax(j)=-lim:kmin(j)=lim:NEXT
   na=0:kx=kx+1:IF kx=510 THEN kx=0:ky=ky+4
   END IF
  IF NOT iab THEN NEXT
 FOR j=0 TO nch1:ly=ky+j*jy:LINE(kx,ly-4)-(kx,ly+4):NEXT
 WEND
Prtn:MENU 8,5,0:IF iab GOTO W2 ELSE RETURN
Atten:GOSUB OpW2:BUTTON 1,1,"Decrease",(5,30)-(70,45)
 BUTTON 2,1,"Increase",(75,30)-(145,45):BUTTON 3,1,"OK",(150,30)-(190,45)
At1:CLS:PRINT" Current atten. = ";ia:WHILE DIALOG(0)<>1:WEND
 id=DIALOG(1):IF id<3 THEN ia=ia-3+id*2:GOTO At1
 FOR i=1 TO 3:BUTTON CLOSE i:NEXT:GOTO W2A
Clip:GOSUB OpW2:BUTTON 1,1,"Decrease",(5,30)-(70,45)
 BUTTON 2,1,"Increase",(75,30)-(145,45):BUTTON 3,1,"OK",(150,30)-(190,45)
Cl1:CLS:PRINT" Current limit = ";lim;" pixels":WHILE DIALOG(0)<>1:WEND
```

```
 id=DIALOG(1):IF id<3 THEN lim=lim-3+id*2:GOTO C11
 FOR i=1 TO 3:BUTTON CLOSE i:NEXT:GOTO W2A
Abort:LOCATE 26,2:PRINT"Aborted.";:iab=-1:RETURN
Quit:WINDOW CLOSE 2:lf=-1:RETURN
```

OBSLoc3

```
'Program OBSLoc3    OBS location from water wave arrival times only **25-Dec-86**
 DEFINT i-n:DEFDBL a-e,s-y:TEXTSIZE 9:TEXTFONT 4
 DIM t(72),ay(72),ax(72),a(2,2):g=57.29578:f=111111!
WWIn:CLS:PRINT"WWData.":f$=FILES$(1,"TEXT"):IF f$="" THEN END ELSE OPEN"I",1,f$
 CLS:PRINT"NavDatI.":OPEN"I",2,FILES$(1,"TEXT"):ays=0:axs=0:m=0:ms=0
 WHILE NOT EOF(1):INPUT#1,ns,t(m),dum
 WHILE ms<ns:INPUT#2,ms,ay(m),ax(m),dep:WEND:IF ms>ns THEN PRINT"Input error"
 ays=ays+ay(m):axs=axs+ax(m):m=m+1:WEND:CLOSE 1:CLOSE 2
 ays=ays/m:axs=axs/m:cp=COS(ays/g)*f:m=m-1:
 FOR i=0 TO m:ay(i)=(ay(i)-ays)*f:ax(i)=(ax(i)-axs)*cp:NEXT
 INPUT"Water depth and speed (m,m/s):",h,c:c=1/c:h=h^2
Init:LINE INPUT"Initial lat,long [,tc]:",in$:j=-1:FOR i=1 TO LEN(in$):IF
MID$(in$,i,1)<>"," GOTO Inxt
 IF j THEN ala=VAL(MID$(in$,1,i-1)):il=i+1:j=0:GOTO Inxt
 alo=VAL(MID$(in$,il,i-il)):u(2)=VAL(MID$(in$,i+1)):n=3:GOTO In2
Inxt:NEXT:alo=VAL(MID$(in$,il)):n=2
In2:n1=n-1:u(0)=(ala-ays)*f:u(1)=(alo-axs)*cp:nit=0
 INPUT"dampling:",dmp
Loop:FOR l=0 TO n1:FOR k=0 TO n1:a(l,k)=0:NEXT:b(l)=0:NEXT:ss=0
 FOR i=0 TO m:d0=u(0)-ay(i):d1=u(1)-ax(i):r=SQR(d0^2+d1^2+h)
 res=t(i)-u(2)-r*c:ss=ss+res^2:IF res>0 THEN PRINT"+";ELSE PRINT" ";
 d(0)=d0*c/r:d(1)=d1*c/r:d(2)=1
 FOR j=0 TO n1:FOR k=0 TO n1:a(j,k)=a(j,k)+d(j)*d(k):NEXT:b(j)=b(j)+res*d(j):NEXT
 NEXT:PRINT:FOR l=0 TO n1:PRINT USING"######.###";u(l);:NEXT:PRINT ss
 IF nit>0 AND ss=>ss1 GOTO Out ELSE GOSUB Inv
 FOR l=0 TO n1:PRINT USING"######.###";b(l);
 v(l)=u(l):u(l)=u(l)+b(l)/dmp:NEXT:PRINT:nit=nit+1:ss1=ss:GOTO Loop
Out:PRINT USING"#####.#####";v(0)/f+ays;v(1)/cp+axs;
 IF n=3 THEN PRINT USING"#####.###";v(2);
 PRINT USING"#####.###";SQR(ss1/(m+1));:PRINT nit-1
 INPUT"",q$:IF q$="" THEN GOTO WWIn ELSE GOTO Init
Inv:FOR j=0 TO n1:amax=0
 FOR i=j TO n1:j1=j+1:IF ABS(a(i,j))>ABS(amax) THEN amax=a(i,j):imax=i
 NEXT:FOR k=j TO n1:IF imax>j THEN tmp=a(j,k):a(j,k)=a(imax,k):a(imax,k)=tmp
 a(j,k)=a(j,k)/amax:NEXT:tmp=b(imax):b(imax)=b(j):b(j)=tmp/amax
 IF j<n1 THEN FOR i=j1 TO n1:FOR k=j1 TO n1:a(i,k)=a(i,k)-
a(i,j)*a(j,k):NEXT:b(i)=b(i)-b(j)*a(i,j):NEXT
 NEXT:FOR j=n-2 TO 0 STEP -1:FOR k=n1 TO j+1 STEP -1:b(j)=b(j)-
a(j,k)*b(k):NEXT:NEXT:RETURN
```

OBSLoc4

```
'Program OBSLoc4    OBS location from water waves **3-Feb-87**
 DEFINT i-n:DEFDBL a-e,s-y:TEXTSIZE 9:TEXTFONT 4
 g=57.29578:f=111111!:pi=3.14159265#:pi2=pi*2:hpi=pi/2
 DIM t(72),az(72),ay(72),ax(72),rt(72),a(3,3),aa(1,1,1),bb(1,1),dss(1,1):n=4
WWIn:CLS:PRINT"WWData.":f$=FILES$(1,"TEXT"):IF f$="" THEN END ELSE OPEN"I",1,f$
 i=LEN(f$):WHILE MID$(f$,i,1)<>".":i=i-1:WEND:f$=MID$(f$,i+1)
 CLS:PRINT"NavDatI.":OPEN"I",2,FILES$(1,"TEXT"):ays=0:axs=0:mm=0:ms=0
 WHILE NOT EOF(1):INPUT#1,ns,t(mm),az(mm):az(mm)=az(mm)/g
 WHILE ms<ns:INPUT#2,ms,ay(mm),ax(mm),dep:WEND:IF ms>ns THEN PRINT"Input error"
 ays=ays+ay(mm):axs=axs+ax(mm):mm=mm+1:WEND:CLOSE 1:CLOSE 2:m1=mm-1
 ays=ays/mm:axs=axs/mm:cp=COS(ays/g)*f
 FOR i=0 TO m1:ay(i)=(ay(i)-ays)*f:ax(i)=(ax(i)-axs)*cp:NEXT
 INPUT"Water depth and speed (m,m/s):",h,c:hh=h^2:c=1/c
Init:INPUT"Initial lat,long,orient'n(deg),tc:",ala,alo,ori,u(3)
 u(0)=(ala-ays)*f:u(1)=(alo-axs)*cp:u(2)=ori/g
Inpd:INPUT"Variance weight ratio, (rad/sec)^2:",vr:INPUT"Damping:",dmp:nit=0
Loop:FOR i=0 TO 1:ss(i)=0:FOR j=0 TO 1:bb(i,j)=0:FOR k=0 TO
1:aa(i,j,k)=0:NEXT:NEXT:NEXT
 FOR j=2 TO 3:FOR i=0 TO 3:a(i,j)=0:NEXT:b(j)=0:NEXT
 FOR i=0 TO m1:y=u(0)-ay(i):x=u(1)-ax(i):dd=x^2+y^2:e=SQR(dd+hh):ce=c/e
 rs=az(i)-hpi+ATN(y/x)+u(2):rt(i)=t(i)-e*c-u(3):res(1)=rt(i)
 WHILE rs<-hpi:rs=rs+pi:WEND:WHILE rs>hpi:rs=rs-pi:WEND:res(0)=rs
 IF rs>0 THEN PRINT"+"; ELSE PRINT" ";
 dss(0,0)=-x/dd:dss(0,1)=y/dd:dss(1,0)=y*ce:dss(1,1)=x*ce
 FOR k=0 TO 1:ss(k)=res(k)^2+ss(k):a(k,2)=a(k,2)-dss(0,k):a(k,3)=a(k,3)+dss(1,k)
 FOR l=0 TO 1:bb(l,k)=res(l)*dss(l,k)+bb(l,k)
 FOR m=0 TO 1:aa(m,k,l)=dss(m,k)*dss(m,l)+aa(m,k,l):NEXT:NEXT:NEXT
 b(2)=b(2)-res(0):b(3)=b(3)+res(1):NEXT:PRINT
 FOR i=0 TO m1:IF rt(i)>0 THEN PRINT"+";ELSE PRINT" ";
 NEXT:PRINT
 FOR k=0 TO 1:b(k)=bb(0,k)+bb(1,k)*vr:a(k,3)=a(k,3)*vr
 FOR l=0 TO 1:a(k,l)=aa(0,k,l)+aa(1,k,l)*vr:a(l+2,k)=a(k,l+2):NEXT:NEXT
 a(2,2)=mm:a(3,3)=mm*vr:b(3)=b(3)*vr:s=ss(0)+ss(1)*vr
```

```
 FOR l=0 TO 3:PRINT USING"######.###"; u(l);:NEXT:PRINT s
 IF nit>0 AND s=>sl GOTO Out ELSE GOSUB Inv
 FOR l=0 TO 3:PRINT USING"######.###";b(l);
 v(l)=u(l):u(l)=u(l)+b(l)/dmp:NEXT:PRINT:nit=nit+1
 FOR i=0 TO 1:ssl(i)=ss(i):NEXT:sl=s:GOTO Loop
Out:v0=v(0)/f+ays:v1=v(1)/cp+axs:v2=v(2)*g
 m4=mm-4:rmss=SQR(ssl(0)/m4)*g:rmst=SQR(ssl(1)/m4)
 PRINT USING"#####.#####";v0;v1;:PRINT USING"#####.###";v2;v(3);
 PRINT USING"####.###";rmss;rmst;:PRINT vr;dmp;nit-1
O1:INPUT"",q$:IF q$="" THEN GOTO WWIn ELSE IF q$="c" GOTO Inpd
 IF q$<>"p" GOTO Init ELSE LPRINT f$;
 LPRINT USING"#####.#####";v0;v1;:LPRINT USING"#####.###";v2;v(3);
 LPRINT USING"####.###";rmss;rmst;:LPRINT"  ";mm;vr:GOTO O1
Inv:n1=n-1:FOR j=0 TO n1:amax=0
 FOR i=j TO n1:j1=j+1:IF ABS(a(i,j))>ABS(amax) THEN amax=a(i,j):imax=i
 NEXT:FOR k=j TO n1:IF imax>j THEN tmp=a(j,k):a(j,k)=a(imax,k):a(imax,k)=tmp
 a(j,k)=a(j,k)/amax:NEXT:tmp=b(imax):b(imax)=b(j):b(j)=tmp/amax
 IF j<n1 THEN FOR i=j1 TO n1:FOR k=j1 TO n1:a(i,k)=a(i,k)-
a(i,j)*a(j,k):NEXT:b(i)=b(i)-b(j)*a(i,j):NEXT
 NEXT:FOR j=n-2 TO 0 STEP -1:FOR k=n1 TO j+1 STEP -1:b(j)=b(j)-
a(j,k)*b(k):NEXT:NEXT:RETURN
```

SegYAtoE/C

```
'Program SegYAtoE  ASCI to EBCDIC SegY header converson *02-Mar-90*
DEFINT i-n:TEXTFONT 4:TEXTSIZE 9
 DIM k(127):FOR i=0 TO 127:READ k(i):NEXT
 DATA 0,1,2,3,55,45,46,47
 DATA 22 ,5,37,11,12,13,14,15
 DATA 16,17,18,59,60,61,50,38
 DATA 24 ,25,63,39,28,29,30,31
 DATA 64,90,127,123,91,108,80,125
 DATA 77,93,92,78,107,96,75,97
 DATA 240,241,242,243,244,245,246,247
 DATA 248,249,122,94,76,126,110,111
 DATA 124,193,194,195,196,197,198,199
 DATA 200,201,209,210,211,212,213,214
 DATA 215,216,217,226,227,228,229,230
 DATA 231,232,233,173,224,189,95,109
 DATA 121,129,130,131,132,133,134,135
 DATA 136,137,145,146,147,148,149,150
 DATA 151,152,153,162,163,164,165,166
 DATA 167,168,169,192,106,208,161,7
Begin:CLS:PRINT"Open combined AscHdr. file for input:":f$=FILES$(1,"TEXT"):IF
f$=""THEN END
 OPEN"I",1,f$:i=LEN(f$):WHILE MID$(f$,i,1)<>":":i=i-1:WEND:dir$=MID$(f$,1,i):PRINT
dir$
 fileopen=0
WHILE NOT EOF(1):LINE INPUT#1,a$
 IF MID$(a$,1,1)="A"THEN
  IF fileopen THEN GOSUB Finish
  ff$="Ebc"+MID$(a$,4):PRINT ff$:f$=dir$+ff$
  OPEN"O",2,f$:CLOSE 2:OPEN"R",2,f$,80:FIELD 2,80 AS out$
  l=0:fileopen=-1
 ELSE
  b$=CHR$(195)+STRING$(79,64)
  FOR i=1 TO LEN(a$):MID$(b$,i,1)=CHR$(k(ASC(MID$(a$,i)))):NEXT
  LSET out$=b$:PUT 2:l=l+1
  END IF
 WEND:GOSUB Finish:CLOSE 1:END
'
Finish:LSET out$=CHR$(195)+STRING$(79,64)
 WHILE l<40:PUT 2:l=l+1:WEND:CLOSE 2:RETURN
```

SegySetup

```
'Program SegySetup **02-Aug-91**WGS84**10-Nov-92**
' - select blocks to be copied and prepare header information for SEG-Y
DEFINT i-n:DEFDBL a-e,l,p-u:TEXTFONT 4:TEXTSIZE 9
 pi=3.141592653589793#:c=180#/pi:pi2=pi*2:pih=pi/2
 a=6378.137#:e=.006739497#:e1=1+e:e2=.00669438#:ec=1-e2
 c1=.01953125#:c2=.046875#:c3=.0146484375#:c4=.03125#
 c5=.0029296875#:c6=.00390625#:c7=3072#
 kc0=ASC("0"):ko0=ASC("A")-kc0-10:tds=86400#
DIM ner(99),idb(12),ll(1,1)
FOR i=1 TO 12:READ idb(i):NEXT:DATA 0,31,59,90,120,151,181,212,243,273,304,334
MENU 8,0,1,"Display":MENU 8,1,1,"Disable":jdf=-1
ON MENU GOSUB Disp:MENU ON
'
Start:PRINT"Open ShotNTLLD.":f$=FILES$(1,"TEXT"):IF f$=""THEN END
 OPEN"I",1,f$
PRINT"Open HdrList.":f$=FILES$(1,"TEXT"):OPEN"I",2,f$
```

```
 LINE INPUT#2,b$:CLOSE 2:OPEN"I",2,f$
 HexDec 14,nbr:nchi=VAL(MID$(b$,18,1)):HexDec 19,nsi
 HexDec 29,kyr:kyr=1900+kyr
 PRINT"Number of blocks per record =";nbr
 PRINT"Number of input channels =";nchi
 PRINT"Nominal sample interval =";nsi;"ms"
 PRINT"Year =";kyr
PRINT"Hit any key and open SegySetupPar.":GOSUB Pause:f$=FILES$(1,"TEXT")
 IF f$<>""THEN
  OPEN"I",3,f$
  INPUT#3,job,lne,ncho,tsi,awo,wsr,trl
  INPUT#3,dobs,dag,lao,loo,faf
  INPUT#3,cc,kd0,khr,kmn,cdr,shdly
  FOR j=0 TO 99:INPUT#3,ner(j):IF ner(j)>0 THEN NEXT
  jp=j-1:CLOSE 3
  PRINT"Job No.";job
  PRINT"Line No.";lne
  PRINT"Number of output channels =";ncho
  PRINT"Sample interval =";tsi;"ms"
  PRINT"Advance window opening =";awo;"s"
  PRINT"Window shift rate =";wsr;"km/s"
  PRINT"Minimum trace length =";trl;"s"
  PRINT"OBS depth =";dobs;"m"
  PRINT"Air-gun depth =";dag;"m"
  PRINT"OBS lat. & long. =";lao;loo;"deg"
  PRINT"Alias filter frequency =";faf;"Hz"
  PRINT"Clock correction =";cc;"s at";kd0;khr;kmn
  PRINT"Clock drift rate =";cdr;"ms/hr"
  PRINT"Shot delay =";shdly;"s"
  PRINT"Time error records:";:FOR j=0 TO jp:PRINT ner(j);:NEXT:PRINT
  GOSUB Pause
 ELSE
  INPUT"Job No.",job
  INPUT"Line No.",lne
  INPUT"Number of output channels =",ncho
  INPUT"Sample interval (ms) =",tsi
  INPUT"Advance window opening (s) =",awo
  INPUT"Window shift rate (km/s) =",wsr
  INPUT"Minimum trace length (sec) =",trl
  INPUT"OBS depth (m) =",dobs
  INPUT"Air-gun depth (m) =",dag
  INPUT"OBS lat. & long. (deg)",lao,loo
  INPUT"Alias filter frequency (Hz) =",faf
  INPUT"Clock correction (s) at (dy-of-yr,hr,mn):",cc,kd0,khr,kmn
  INPUT"Clock drift rate (ms/hr) =",cdr
  INPUT"Shot delay (s) =",shdly
  PRINT"Time error records:"
  FOR j=0 TO 99:INPUT"",ner(j):IF ner(j)>0 THEN NEXT
  jp=j-1
  f$=FILES$(0,"Save in SegySetupPar."):IF f$<>""THEN
   OPEN"O",3,f$
   PRINT#3,job;lne;ncho;tsi;awo;wsr;trl
   PRINT#3,dobs;dag;lao;loo;faf
   PRINT#3,cc;kd0;khr;kmn;cdr;shdly
   FOR j=0 TO jp:PRINT#3,ner(j);:NEXT:PRINT#3," 0"
   CLOSE 3
   END IF
 END IF
 isi=tsi*1000#
 nspb=2040/nchi:tbl=nspb*tsi/1000#
 nbs=INT(trl/tbl):nbmax=nbr-nbs:nbl=nbs+1:nsmp=nspb*nbl:trec=nbr*tbl
 pa=lao/c:looc=loo/c
 tlat=lao*36000#:tlon=loo*36000#
 t0=(khr*60#+kmn)*60:cdr=cdr/3600000&:j=0
 PRINT"Sampling interval =";isi;"µs"
 PRINT"Samples per block =";nspb
 PRINT"Block length =";tbl;"s"
 PRINT"Blocks per trace =";nbl
 PRINT"Samples per trace =";nsmp
 PRINT"Limits of starting block = ";1-nbs;nbmax
 PRINT"OBS lat-long =";:PRINT USING"#########";tlat;tlon
 PRINT"t0, cdr =";t0;cdr
 IF (kyr\4)*4=kyr THEN
  IF idb(3)=59 THEN FOR i=3 TO 12:idb(i)=idb(i)+1:PRINT idb(i);:NEXT:PRINT
 ELSE
  IF idb(3)=60 THEN FOR i=3 TO 12:idb(i)=idb(i)-1:PRINT idb(i):NEXT:PRINT
  END IF
'
f$=FILES$(0,"Output: SegyPar."):jf=f$<>""
IF jf THEN
 OPEN"O",3,f$
```

```
 PRINT#3,job;lne
 PRINT#3,dobs;dag
 PRINT#3,USING"##########";tlon;tlat
 PRINT#3,nchi;ncho;isi;nbl;nsmp
 PRINT#3,faf;kyr
 END IF
'
first=-1
WHILE NOT EOF(1):INPUT#1,ns,ids,ihs,ms,ss,las,los,idw
 ts=(ihs*60#+ms)*60#+ss+shdly
'compute shot distance and azimuth
 pb=las/c:dl=los/c-looc
 spa=SIN(pa):cpa=COS(pa):spb=SIN(pb):cpb=COS(pb):sdl=SIN(dl):cdl=COS(dl)
 spas=spa^2:b1=a/SQR(1-e2*spas):b2=a/SQR(1-e2*spb^2)
 den=((spb/spa/e1+b1/b2*e2)*cpa/cpb-cdl)*spa
 az=ATN(sdl/den)+(1-SGN(den))*pih:IF az<0 THEN az=az+pi2
 caz=den/SQR(den^2+sdl^2):cc2=(cpa*caz)^2
 e0=(cc2+spas)*e:e01=1+e0:e01r=SQR(e01):e02=e0^2
 den=cpa*caz*e01r:u1=ATN(spa/den)+(1-SGN(den))*pih
 den=(cdl*caz-sdl*spa*SIN(az))*b2*cpb*e01r
 u2=ATN((b1*spa+e01*(b2*spb-b1*spa)*ec)/den)+(1-SGN(den))*pih
 du=u2-u1:IF du<0 THEN du=du+pi2
 ds=(((c1*e0-c2)*e0+.25#)*e0+1)*du
 ds=ds-((c3*e0-c4)*e0+.125#)*e0*(SIN(2*u2)-SIN(2*u1))
 ds=ds+(c5*e0-c6)*e02*(SIN(4*u2)-SIN(4*u1))
 ds=(ds-e02*e0/c7*(SIN(6*u2)-SIN(6*u1)))*b1/e01*SQR(e*cc2+1)
 az=az*c
'find data window
 tt=ts+ds/wsr-awo
' PRINT ns;ids;ts;las;los:PRINT idw;ds;az;tt
 tlat=las*36000#:tlon=los*36000#
 dm=ds*1000:az=az*100
 IF NOT first GOTO td
InH:IF EOF(2) GOTO Ex ELSE LINE INPUT#2,b$
 HexDec 7,n1: HexDec 9,n2: nr=n2*256+n1
 HexDec 29,kyr:HexDec 32,mo:HexDec 34,kdy:HexDec 37,khr:HexDec 39,mn
 HexDec 42,ksc:HexDec 44,kds:sec=ksc+kds/10!
 idr=idb(mo)+kdy
 tr=(khr*60#+mn)*60#+sec
 IF nr=ner(j) THEN tr=tr+.3#:j=j+1:PRINT"*";
 td=tr-t0+(idr-kd0)*tds:tc=td*cdr+cc:tq=tr+tc
 PRINT nr;:PRINT USING"######.###";td;tc
 iffid=nr*100
td:tdd=(ids-idr)*tds:dt=tt-tq+tdd
 IF jdf THEN PRINT USING"#######";dt
 IF dt>trec  GOTO InH ELSE nb=INT(dt/tbl)+1
 IF nb>-nbs THEN
  IF nb>0 THEN
   tf=tq+(nb-1)*tbl
   IF nb>nbmax THEN kbl=nbr+1-nb ELSE kbl=nbl
  ELSE
   tf=tq:kbl=nb+nbs:nb=1
   END IF
  ldt=(tf-ts-tdd)*1000#
  idf=idr:IF tf<0 THEN tf=tf+tds:idf=idf-1 ELSE IF tf>=tds THEN tf=tf-
tds:idf=idf+1
  ihf=INT(tf/3600#):tf=tf-ihf*3600#:imf=INT(tf/60#):tf=tf-imf*60#
  isf=INT(tf):msf=(tf-isf)*1000#:IF msf=1000 THEN
   msf=0:isf=isf+1:IF isf=60 THEN
    isf=0:imf=imf+1:IF imf=60 THEN
     imf=0:ihf=ihf+1:IF ihf=24 THEN ihf=0:idf=idf+1
     END IF
    END IF
   END IF
  IF first THEN
   dc=0:first=0
  ELSE
   dc=dm-dml
   IF(db>=0 AND dc>=0)OR(db<0 AND dc>0 AND (-db)<dc)OR(db>0 AND dc<0 AND db>-
dc)THEN dml=-dml
   GOSUB Out
   END IF
  IF jdf THEN
   PRINT USING"####";nr;nb;:PRINT USING"###";kbl;
   PRINT USING"######";iffid;ns;:PRINT"  |";
   END IF
  IF jf THEN
   PRINT#3,USING"####";nr;nb;:PRINT#3,USING"###";kbl;
   PRINT#3,USING"######";iffid;ns;
   END IF
  iffid=iffid+1:dml=dm:idwl=idw:tlonl=tlon:tlatl=tlat
```

```
   ldtl=ldt:idfl=idf:ihfl=ihf:imfl=imf:isfl=isfl:msfl=msf:azl=az:db=dc
   IF nb>nbmax GOTO InH
   END IF
  WEND
 Ex:IF dc>0 THEN dml=-dml
  GOSUB Out:CLOSE 1:CLOSE 2:IF jf THEN CLOSE 3
  GOTO Start
 '
 Disp:IF MENU(0)<>8 THEN RETURN ELSE jdf=-1-jdf
  IF jd THEN MENU 8,1,1,"Disable" ELSE MENU 8,1,1,"Enable"
  RETURN
 Pause:WHILE INKEY$="":WEND:RETURN
 Out:IF jdf THEN
   PRINT USING"########";dml;:PRINT USING"#####";idwl;
   PRINT USING"#########";tlonl;tlatl;:PRINT USING"######";ldtl;:
   PRINT USING"####";idfl;:PRINT USING"###";ihfl;imfl;isfl;
   PRINT USING"####";msfl;:PRINT USING"######";azl
   END IF
  IF jf THEN
   PRINT#3,USING"########";dml;:PRINT#3,USING"#####";idwl;
   PRINT#3,USING"#########";tlonl;tlatl;:PRINT#3,USING"######";ldtl;:
   PRINT#3,USING"####";idfl;:PRINT#3,USING"###";ihfl;imfl;isfl;
   PRINT#3,USING"####";msfl;:PRINT#3,USING"######";azl
   END IF
  RETURN
 SUB HexDec(ip,idec) STATIC:SHARED b$,kc0,ko0
  FOR i=ip TO ip+1
   k=ASC(MID$(b$,i,1))-kc0:IF k>9 THEN k=k-ko0:IF k>32 THEN k=k-32
   IF i=ip THEN kk=k ELSE idec=kk*16+k
  NEXT:END SUB
```

ShotTimeCheck

```
'Program ShotTimeCheck  **20-May-91**
DEFINT i-n:DEFDBL t:TEXTSIZE 9:TEXTFONT 4
OpFl:CLS:PRINT"Open shot-time file.":f$=FILES$(1,"TEXT")
 IF f$=""THEN END ELSE OPEN"I",1,f$
 PRINT"Open output file.":f$=FILES$(0,"Output"):lf=f$<>""
 IF lf THEN OPEN"O",2,f$
 INPUT"Nominal shot interval (sec) =",si
 INPUT"Tolerance (sec) =",ds:tl=0
WHILE NOT EOF(1):INPUT#1,a$:IF LEN(a$)>16 THEN
 kd=VAL(MID$(a$,2,3))
 t=(VAL(MID$(a$,6,2))*60+VAL(MID$(a$,9,2)))*60#+VAL(MID$(a$,12,6))
 IF tl=0 THEN
  id=kd
 ELSE
  IF kd>id THEN t=t+86400#
  dt=t-tl:IF ABS(dt-si)>ds THEN
   PRINT a$;:PRINT USING"####.###";dt
   IF lf THEN PRINT#2,a$;:PRINT#2,USING"####.###";dt
   END IF
 END IF
 tl=t
 END IF
 WEND:CLOSE 1:CLOSE 2
 PRINT"Hit any key to continue."
 WHILE INKEY$="":WEND:GOTO OpFl
```

WTAPlot

```
'Program WTAPlot    Water-wave travel time and azimuth plot
' **23-Dec-86** Rev. 15-Aug-92**
 DEFINT i-l:DEFDBL u-v
 g=57.29578:f=111111!:pi=3.141593:pi2=pi*2:zs=250/360:gs=g*zs
 WINDOW 2,,(1,20)-(500,325),3:TEXTSIZE 9:TEXTFONT 4
Fls:PRINT"WWData.:":f$=FILES$(1,"TEXT"):IF f$=""THEN END ELSE OPEN"I",1,f$:ll=-1
 INPUT"Time range:",tb,tm:ts=250/(tm-tb):tbb=tb+275/ts
 INPUT"Maximum x:",xm:xs=450/xm
 PRINT"navig.:":ff$=FILES$(1,"TEXT"):nav=ff$<>"":IF NOT nav GOTO Plot ELSE
OPEN"I",2,ff$
 INPUT"OBS lat,long,depth,vel,orie'n,tc:",v0,u0,d0,vel,a0,tc
 c=COS(v0/g)*f:la=(a0=>0):ar0=a0/g
Plot:CLS:IF nav THEN MOVETO 10,10:PRINT v0;u0;d0;vel;a0;tc
 MOVETO 10,300:PRINT f$;:m=0
 FOR t=tb TO tm STEP .5:l=275-ts*(t-tb)
 MOVETO 10,l+4:PRINT USING"#.#";t;:LINE(35,l)-(40,l):NEXT
 FOR i=0 TO 360 STEP 90:l=275-i*zs:LINE(450,l)-(455,l):MOVETO 460,l+4:PRINT
i;:NEXT
 WHILE NOT EOF(1):INPUT#1,n,t,a:y=ts*(tbb-t):z=275-a*zs
 IF NOT nav GOTO P1 ELSE WHILE m<n:INPUT#2,m,v,u,d:WEND
 IF m>n THEN PRINT"Input error";n;m
```

```
 v=(v0-v)*f:u=(u0-u)*c:w=ATN(u/v):IF v<0 THEN w=w+pi
 w=w-ar0:WHILE w<0:w=w+pi2:WEND:WHILE w>pi2:w=w-pi2:WEND
 v=ts*(tbb-SQR(v^2+u^2+d0^2)/vel-tc):w=275-w*gs
P1:IF l1 THEN n1=n:l1=0:IF nav THEN v1=v:w1=w:x1=50
 x=xs*(n-n1)+50:CIRCLE(x,y),1:LINE(x-1,z-1)-(x+1,z+1):LINE(x-1,z+1)-(x+1,z-1)
 IF nav THEN LINE(x1,v1)-(x,v):v1=v:IF la THEN LINE(x1,w1)-(x,w):w1=w
 x1=x:WEND:CLOSE 1:IF nav THEN CLOSE 2
 WHILE INPUT$(1)="":WEND:CLS:GOTO F1s
```