5-1991

# Inverted Echo Sounder Data Processing Report

Erik Fields

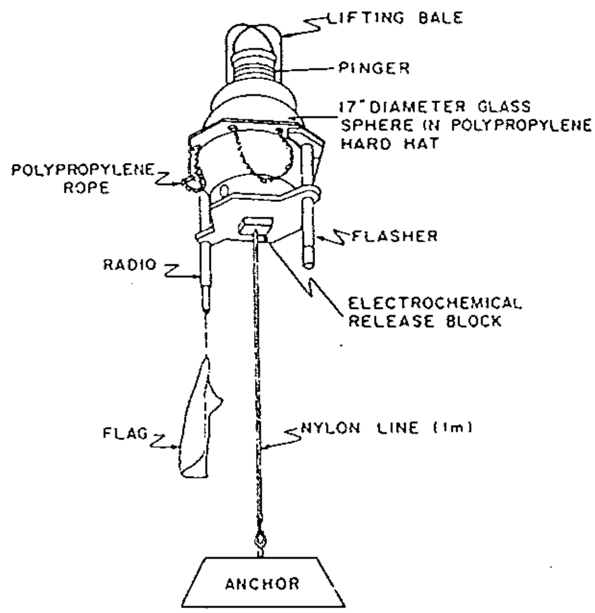
Karen L. Tracey
*University of Rhode Island*, krltracey@uri.edu

D. Randolph Watts
*University of Rhode Island*, randywatts@uri.edu

GRADUATE SCHOOL OF OCEANOGRAPHY
UNIVERSITY OF RHODE ISLAND
NARRAGANSETT, RHODE ISLAND

# Inverted Echo Sounder Data Processing Report

GSO Technical Report No. 91-3





by

Erik Fields, Karen Tracey, and D. Randolph Watts


May

1991


This research program has been sponsored by the National Science Foundation under grant number OCE87-17144 and by the Office of Naval Research under contracts N00014-909J-1568 and N00014-90J-1548.

Abstract

The Inverted Echo Sounder (IES) is an instrument that acoustically monitors the depth of the main thermocline from a moored position one meter above the ocean floor. Additionally, the IESs can be equipped to measure both pressure and temperature. The standard steps for processing IES data are documented here. The effect and purpose of each step are discussed followed by a description of how to apply the computer programs that constitute the step. The FORTRAN and MATLAB codes are also supplied.

# Contents

# List of Figures



# List of Tables

# 1 Overview of IES Processing

The IES is an instrument that acoustically monitors the depth of the main thermocline from a moored position one meter above the ocean floor (Chaplin and Watts 1984). IESs are typically configured to emit a set of twenty-four consecutive 10 KHz pings at 10 sec intervals every half hour. The time required for each ping to reach the surface and return is recorded on a digital cassette tape within the instrument. If an IES also measures bottom pressure and temperature (a PIES), these quantities are also written to tape.

All processing steps have been done on MicroVAX II and MicroVAX III computers. The data are processed with a series of FORTRAN and MATLAB routines specifically developed for the IES. The steps are outlined below and schematically illustrated in Figure 2.Figures 1 and 3 illustrate the more

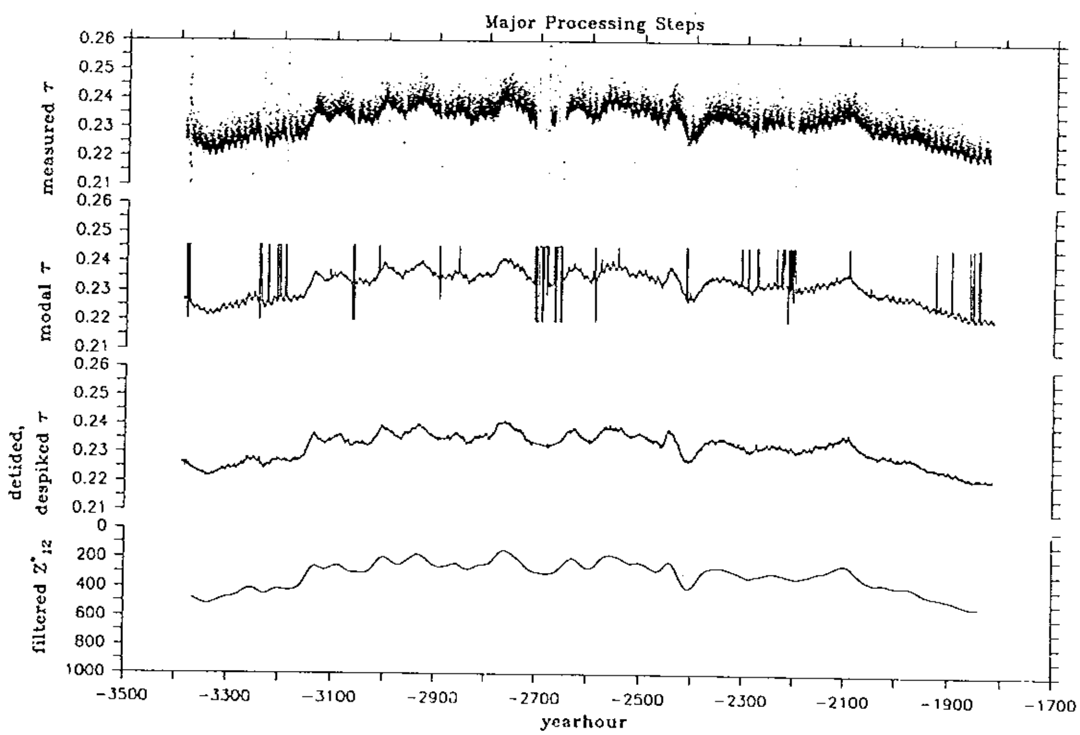



Figure 1: An IES subrecord is plotted at several processing stages to illustrate some of the major steps. The upper panel is the time series of the individual $\tau$'s in seconds. Storms often result in periods of high scatter or a lack of returning echoes. The second plot shows the time series after a single representative $\tau$ is found for each burst of $\tau$'s . The spikes are easily identified and removed, and the tidal signal reduced (third plot). The final plot is the thermocline depth as represented by $\tau$ calibrated to $Z_{12}$ (in meters).

visibly noticeable steps for travel time ($\tau$ ) and pressure. With exception of SDR and BUNS, which are VAX specific, all programs use standard code and could be run in other computing environments.

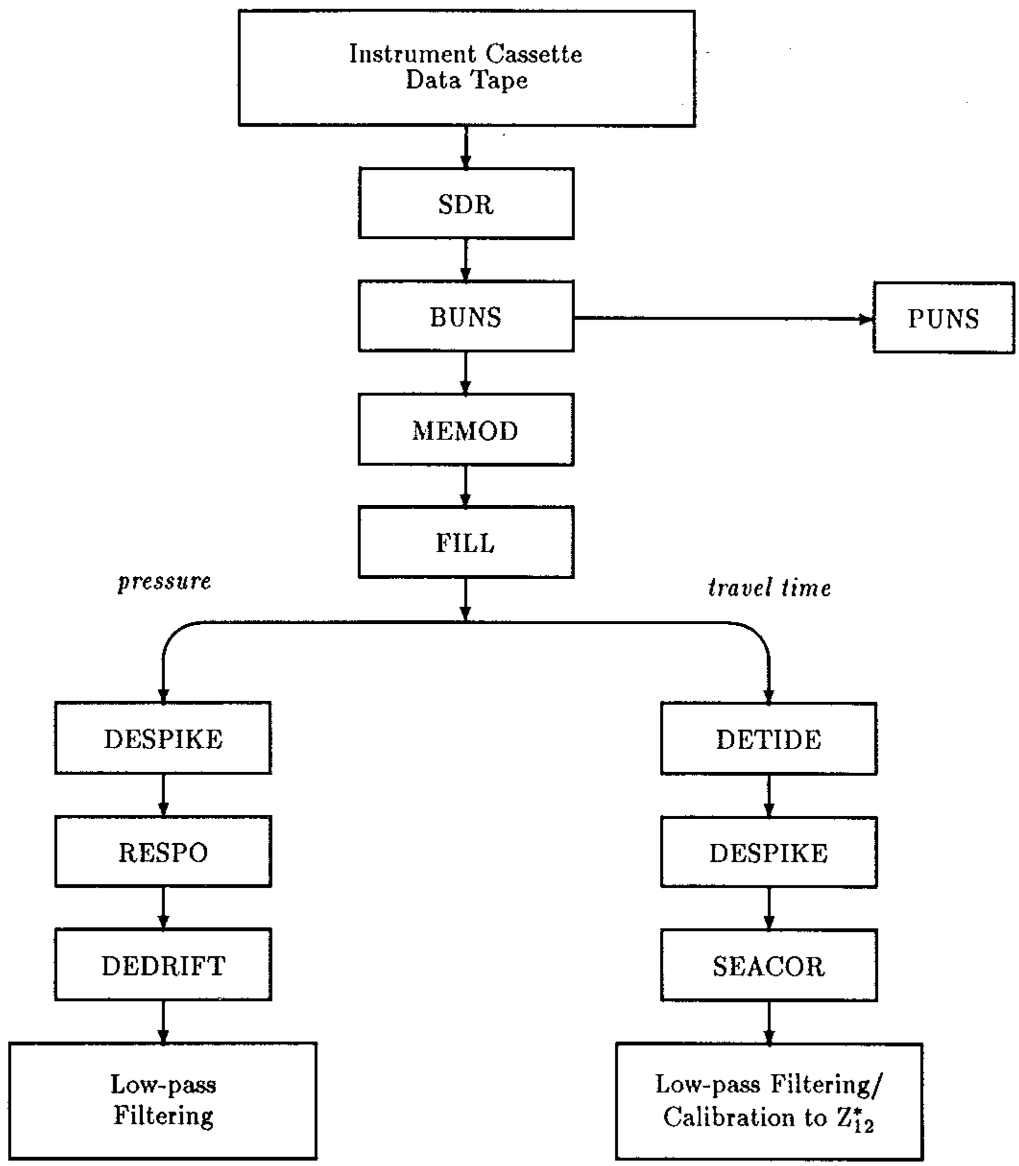



Figure 2: IES Data Processing Flowchart

**Summary of Steps**

**RAW DATA CASSETTES** : The cassette, which is recorded within the IES, contains the counts associated with travel time, pressure, and temperature measurements as a series of integer words of varying lengths.

**SDR** : This program controls the Sea Data Reader, which transfers the data from cassettes directly to unformatted binary files on the MicroVAX.

**BUNS** : Here the series of integer words of varying lengths are converted to standard length 32-bit integer words and are written to ASCII files.

**PUNS** : Integer listings and histograms of the travel times are generated to provide an initial look at data quality and travel time distributions. The histogram is used to determine the limits for maximum and minimum acceptable travel times for an initial windowing operation in the following step. The listings are used to establish the first (after launch) and last (before recovery) 'on bottom' samples essential for determining the time base.

**MEMOD** : At this stage, the time base is established and, after several windowing operations, a single representative $\tau$ is estimated from the burst of $\tau$'s. Travel time, pressure and temperature counts are converted to units of seconds, decibars, and °C respectively.

**FILL** : Proper incrementation of the time base is enforced here. Missing samples are inserted using interpolated values. For PIESs, the temperature and the pressure are each written to separate files with the appropriate time bases.

**DETIDE** : From user-supplied tidal constituents (specific to each site), the tidal contribution to the travel time is estimated and removed.

**DESPIKE** : If present, spikes are identified and replaced with interpolated values.

**SEACOR** : The effects on travel time from seasonal warming and cooling of the surface layers are removed.

**RESPO** : The tides are removed from the pressure records using Response Analysis (Munk and Cartwright, 1977).

**DEDRIFT** : If present, long-term drift in pressure is estimated and removed. Drifts are typically associated with variation in the properties of the sensor crystal over long time-scales or slight imperfection in the IES master clock.

**LOW-PASS FILTERING** : A 2nd-order 40-hr low-pass Butterworth filter is applied forward and backwards. The smoothed series are subsampled at six hour intervals centered on 0000Z, 0600Z, 1200Z, and 1800Z (UT). During this step, travel time is calibrated to $Z_{12}$.

## Travel Time

Variations in the travel times have been shown to be proportional to variations in the thermocline depth in the Gulf Stream region (Watts and Rossby, 1977). For practical purposes the main thermocline depth can be represented by the depth of the 12°C isotherm ($Z_{12}$) as it is situated near the highest temperature gradient of the main thermocline and correlates well with $\tau$ (Rossby, 1969; Watts and Johns, 1982).

In previous studies, $Z_{12}$ was obtained directly from the XBT cast. However, a new method has been developed which takes advantage of the integrative nature of the travel time measurement to give a more representative measure of the thermocline depth. The new measure, $Z_{12}^{*}$, is less susceptible to small, vertical-scale perturbations (i.e., internal waves) in the water column than the single-point measurement, $Z_{12}$. This method consists of calculating $Q$, the 'heat content' ($\int_{250m}^{750m} T dz$) for each calibration XBT cast; then using $Q$ to determine $Z_{12}^{*}$ from an empirical curve relating $Z_{12}$ and $Q$. The curve was established using over 5000 XBT casts in the Gulf Stream region (from NODC archives).

At each IES site, XBTs are taken in order to determine the IES's calibration coefficient (B) necessary to convert travel time into thermocline depth according to the relation: $Z_{12}^{*} = M\tau + B$. The proportionality constant (M) was determined from regressions of all calibration pairs ($Z_{12}^{*}, \tau$) from 1987 to 1990. The regressions showed that the constant value M=–19,800 m/sec was appropriate for all the IESs in the Gulf Stream region. (Hereafter $Z_{12}$ is synonymous with $Z_{12}^{*}$)

The low-pass filtered travel time records are scaled to the thermocline depths. Since $\tau$ is resolved to 0.1 msec, $Z_{12}$ is therefore resolved to ±2 m. However, the accuracy of the offset parameter B is estimated to be ±19 m for most records (judged from the agreements between the calibration XBTs taken at each site).

## Temperature

The thermistor's main purpose is to correct the pressure values for the temperature sensitivity of the transducer. The thermistor is inside the instrument, on the pressure transducer, rather than in the water. However, it provides accurate bottom temperature measurements once the probe has reached equilibrium with the surrounding water. (The measured bottom-temperature fluctuations are effectively low-pass filtered with a two-to-four hour e-folding equilibrium time). The first 24 half-hourly points are dropped prior to filtering, since the temperature takes twelve hours to reach equilibrium within 0.001°C. The accuracy of the temperature measurements is about 0.1°C, and the resolution is 0.0002°C.

## Bottom Pressure

Digiquartz pressure sensors manufactured by Paroscientific Incorporated are used to measure bottom pressure. All pressure measurements are corrected for the temperature sensitivity of the transducer. The measured bottom pressure is dominated by the tide; however, for some of the instruments, the pressure also drifts, O(0.1 dbar $yr^{-1}$), monotonically with time. Processing of the pressure measurements includes removing the long-term drift and tides. Figure 3 illustrates the detiding and filtering of a pressure subrecord.

Response Analysis (Munk and Cartwright, 1977) is used to determine the tidal pressure signal.

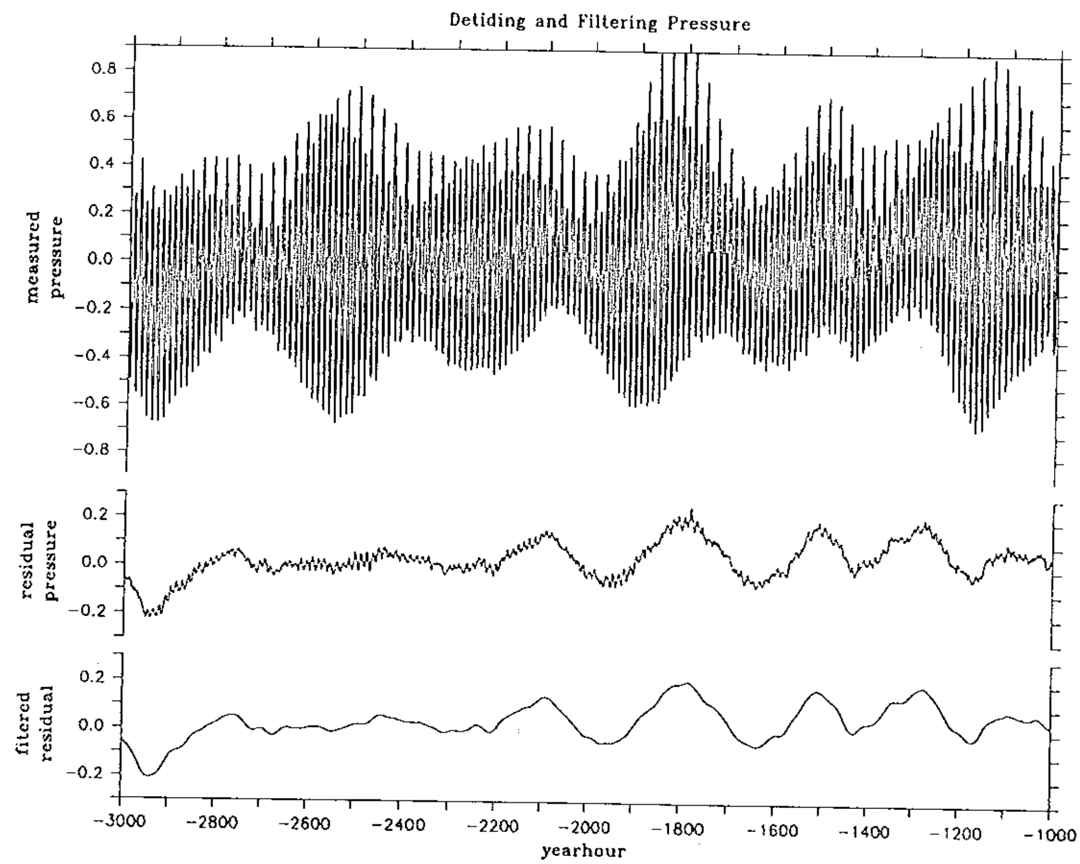



Figure 3: The detiding and filtering of a pressure are illustrated above. Pressure is offset by its mean for the entire record of 4552 dbars.

The predicted tides are then removed from the pressure records.

The pressure records are dedrifted in the manner developed by Watts and Kontoyiannis (1990) who have examined pressure sensor drift and performance. The rate of drift decays with time and is best approximated by an exponential function of the form[1],

$$Drift = Ae^{-\lambda t} + B.$$

A design matrix for the nonlinear least-squares fit would be composed of ($e^{-\lambda t_i}$, 1). The overdetermined set of equations is solved for coefficients $A$ and $B$. These coefficients are found subject to the minimization of the rms error of the fit as a function of the decay rate, $\lambda$. Minimization is accomplished using the method of parabolic extrapolation and golden sections (Press et al., 1988) to optimally search for $\lambda$ with a minimum of function evaluations (fits). The first 12 hours of pressure are ignored since the crystal's temperature equilibrates during that period. The drift curves are usually found from two-hourly subsampled records for computational simplicity. At a later stage, comparison of geostrophic currents (calculated from adjacent dedrifted pressure sensors) and nearby current meters will be used to verify the dedrift procedure's success.

The half-hourly pressures are resolved to 0.001 dbar and the mean pressure is accurate to within 1.5 dbar.

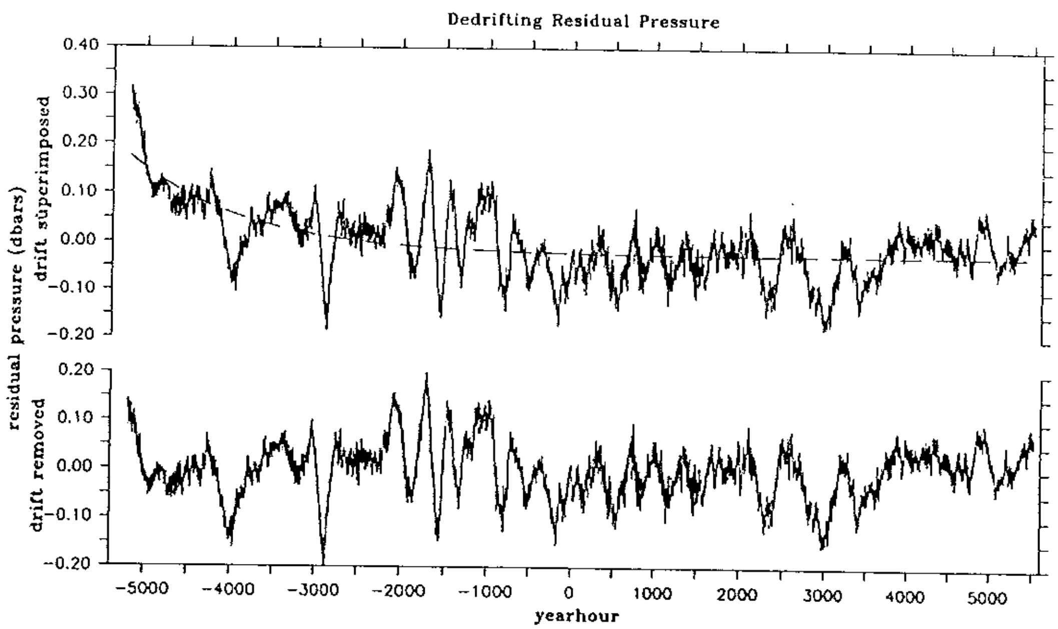



Figure 4: The residual pressure before and after the subtraction of the estimated drift (dashed).

[1]When justified, a linear term is included.

# 2 PROGRAM DESCRIPTIONS

## 2.1 SDR

SDR abbreviates Sea Data Reader. SDR is a program that instructs a Sea Data model-12A Reader to read a four-track cassette tape from an IES and transcribe the phase-encoded data into a binary data file (on the MicroVax machine). At the end of a read (signaled by entering '`<ctrl> z`' from the keyboard), SDR provides statistics of the read: the accumulated number of good and bad records, the number of occurrences of each of the different error conditions, and the distribution of the parity errors across the four tracks.

The output file is composed of unformatted blocks of 512 bytes organized into words (2 bytes). Each word has the format displayed in Table 1.

Table 1: The SDR word format depends on the type of word. Bit 15 indicates whether the word is a data or message word. Here 'PE' stands for parity error, 'cpr' stands for the number of 4-bit characters per record, and low=0, hi=1.

| bit | description | | |
|---|---|---|---|
| | **data word** | **message word** | |
| 0 | data line 0 | PE track 1 | low indicates a PE |
| 1 | data line 1 | PE track 2 | low indicates a PE |
| 2 | data line 2 | PE track 3 | low indicates a PE |
| 3 | data line 3 | PE track 4 | low indicates a PE |
| 4 | data line 4 | | |
| 5 | data line 5 | low signal | hi if low signal occurred |
| 6 | data line 6 | short record | hi if short record occurred |
| 7 | data line 7 | parity error | hi if PE occurred |
| 8 | last character | | hi indicates last data word of record |
| 9 | file gap | | hi after 28 consecutive data-free words |
| 10 | word length (lsb) | | lsb of cpr |
| 11 | word length (msb) | | msb of cpr |
| 12 | | | |
| 13 | | | |
| 14 | overrun | | hi if scans were missed |
| 15 | word type | | hi if message word |

To run SDR a user must have PFNMAP and CMKRNL process privileges (on the VMS operating system). The DCL command "`define SDR :==$sdr.exe`" allows SDR.FOR to be run with qualifiers. These qualifiers can only be entered from the command line. In the following example, the call "`SDR /?`" displays the possible qualifiers (note `RWATTS}` is the system prompt):

```
RWATTS } sdr/?
SDR  X1.05  SEADATA Tape Reader Data Dump Program
May 19 1988 18:28:28
usage: SDR (filename) (switches)
Switches:
                /B   - Buffer numbers displayed
                /Cn  - Characters per record
                /D   - Debug mode
```

```
                /N    - No statistics generated
                /P    - Prompt for device characteristics
      Switches must be separated by spaces.
RWATTS }
```

If no qualifiers are used SDR will prompt for the number of four-bit characters per record (excluding preamble and longitudinal check characters, LCC). The number of characters per record is the same as was wired on the control card in the IES's recorder.

The cassette reader must be set properly for data to be read with a minimum of errors.

| SWITCH | SETTING |
|---|---|
| mode | counter |
| density | 800 bpi |
| control | local |
| output | computer |
| speed | 7.5 ips |
| data | data |

The MASTER GAIN should be set about 60 and the THRESHOLD to 20%. The TRACK GAIN ADJUST knobs should point to about 2 o'clock. With the METER MONITOR switch set to VCO the meter's needle should point to 100%.

The amplifier gain must be checked for each of the four channels. In order to have a calibrated flux detection threshold, the signal levels for each channel should be situated between 100% and 120% when a cassette is being read. The signal levels of the individual channels are checked by switching the METER MONITOR switch from VCO to each of the four channel numbers. All channels are adjusted simultaneously with the MASTER GAIN control, and individual channels may be adjusted using the corresponding TRACK GAIN ADJ potentiometer for that channel.

Several readings should be made of a cassette's contents and the one with the least errors selected for further processing. Below an example of a read session is listed. The first command defines the symbol SDR, which runs the program. The program prompts for the number of 4-bit characters per record and a file prefix to be added to the extension '.sdr' for naming the binary output file. When the tape is not being read or when the final file gap is reached the message `%SDR-I-NODATA, No data is being received from the reader` is displayed. The entry `<ctrl>Z` closes the output file and displays the statistics on the screen. Of the two readings below, the second has more "good" records, fewer parity errors, and fewer overruns flags.

```
RWATTS } sdr :==$rwatts$dua0:[cruise.sdr]sdr.exe
RWATTS } sdr
SDR  X1.05  SEADATA Tape Reader Data Dump Program
May 19 1988 18:28:28
I/O section mapped 75800 to 759FF.
Characters per record? 86
Data file name <.SDR>? sdr_test
```

```
Logging data to file RWATTS$DUA0:[CRUISE.SDR]SDR_TEST.SDR;1

%SDR-I-NODATA, No data is being received from the reader.

%SDR-I-NODATA, No data is being received from the reader.

^Z
Data file RWATTS$DUA0:[CRUISE.SDR]SDR_TEST.SDR;1 closed.

Records -   Good: 15577 (99.1%)   Bad: 135 (0.9%)   Total: 15712
Messages: 15717   File Gaps: 3   Overruns: 117   Parity Errors: 4
Parity Errors -
Track 1: 3
Track 2: 4
Track 3: 3
Track 4: 2

RWATTS }
RWATTS } sdr
SDR  X1.05  SEADATA Tape Reader Data Dump Program
May 19 1988 18:28:28
I/O section mapped 75800 to 759FF.
Characters per record? 86
Data file name <.SDR>? sdr_test_run2

Logging data to file RWATTS$DUA0:[CRUISE.SDR]SDR_TEST_RUN2.SDR;1

%SDR-I-NODATA, No data is being received from the reader.

^Z

Data file RWATTS$DUA0:[CRUISE.SDR]SDR_TEST_RUN2.SDR;1 closed.

Records -   Good: 15592 (99.2%)   Bad: 120 (0.8%)   Total: 15712

Messages: 15716   File Gaps: 3   Overruns: 115   Parity Errors: 3
Parity Errors -
Track 1: 3
Track 2: 3
Track 3: 3
Track 4: 1
```

## 2.2 BUNS_AUG89.FOR and BUNS_ENGIN_AUG89.FOR

The purpose of this program is to create standard length words from a series of integer words of varying lengths. The input is a string of several 'N' bit words, where N ranges between 1 and 31. The output is a series of 32-bit computer words, which contain the 'N' bit string of each word in the least significant position. Padding to the left with zeros is done wherever necessary. The length N of each word to be 'decoded' is supplied by the user and is contained in a control file.

The input bit strings are read from a file created by SDR. Although the basic procedures used in this program could decode any string of bits, there are a few statements which make it specific to be run on a microVAX with a file created by SDR. The output is written, one sampling period at a time, to a disk file. The user specifies the output format, either binary or 13I9, in the control file.

The FORTRAN source code is listed in Section 3.1. The user supplied parameters are described in detail below and two example control files follow.

### CONTROL FILE

This file is composed of a series of parameter lines which are identical in format (A2,3X,10I5). Each line is composed of a character string, IDI, and an integer array, IVALS.

IDI, (IVALS(I),I=1,10)

where:

**IDI** - (CHARACTER*2) IDI is a string that identifies the type of parameters which follow in the array IVALS. IDI has possible values of 'NW', 'WL', 'SV', 'US', and 'WF', which stand for 'number of words', 'word length', 'special value', 'unspan', and 'write format'.

**IVALS(10)** - (INTEGER*4) IVALS contains input parameters of type specified by IDI. The meaning of each element of the array is explained below.

---

### Parameters in the Control File

**if IDI='NW'**

**IDI = 'NW'** this indicated that a 'number of words' array follows. This 'NW' group indicates the number of integer words pertaining to one sampling period which are to be decoded.

**IVALS(1) = NWDS** Total number of non-negative, non-zero words listed on the 'WL' lines.

**IVALS(2) = NSECT** The number of cassette records needed to hold all the data from one sampling period. This should be equal to the number of -1 values on the 'WL' lines.

**If IDI = 'WL'**

**IDI='WL'** 'WL' denotes a 'word length' array, which gives the length in bits of each word to be decoded into a 32-bit word. Typically, several 'WL' lines are required specify all the word lengths.

**IVALS(1-10) = DECODE(1-10)** Array of word lengths to be decoded. All zero values are ignored. The end of the cassette record is flagged by -1.

**If IDI = 'SV'**

**IDI = 'SV'** 'SV' denotes a 'special value' array. This array signals that some of the words are expected to have specific values.

**IVALS(I) = TESTW** The word number which is to be tested for a specific value.

**IVALS(I+1) = TESTV** The value that TESTW is expected to have. It is ignored if its value is either negative or zero.

**If IDI = 'US'**

**IDI = 'US'** this is the 'unspan' array. This indicates that the bits associated with a single data value actually span two cassette records and these need to be joined to form a single 32-bit word.

**IVALS(1) = ITHROT** If it is less than or equal to zero, all words will be processed. If greater than zero, the corresponding word will not to be converted to a 32-bit word and its value will be lost.

**IVALS(I), IVALS(I+1)** ; where I > 1. This pair of words are to be unspanned. The bits stored in IVALS(I) are of higher order than those in IVALS(I+1). If the value of these are zero, they are ignored.

**If IDI = 'WF'**

**IDI = 'WF'** this is the 'write format' line. This is used to determine the format of the output data file.

**IVALS(1) = KW1FMT** If = 0, output is binary. If = 1, output will have the format (13I9).

## EXAMPLE CONTROL FILES

Two examples of control files are listed in Table 2. The first one, MOD_92CPR.CTRL, is relatively simple with one cassette record corresponding to one sample period. Thus there are no words which need to be unspanned. It is used with IESs which have 92 4-bit data characters (368 bits) recorded on each cassette record. In this example, the data included in the 368 bits are one 16-bit sequence number word, twenty-four 13-bit travel time words, one 24-bit pressure word, and one 16-bit temperature word. The output data set will be written in binary format.

The second one, MOD_82CPR.CTRL, is more complex with the data from one sampling period spanning three cassette records. The 'SV' card indicates that there are three words which are expected to have specific values: Word 1 is expected to be zero. Words 24 and 48 are both expected to contain the value 1. The 'US' card indicates that the very first word is not to be converted to a 32-bit word, and its contents will not be saved in the output data set. Additionally, the 11 bits of word 23 and the 7 bits of word 25 are to be combined to form a single data word that will be 18 bits

Table 2: Two examples of control files for BUNS_AUG89.FOR. See text for explanations.

MOD_92CPR.CTRL

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| NW | 27 | 01 | | | | | | | | |
| WL | 16 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 |
| WL | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 | 13 |
| WL | 13 | 13 | 13 | 13 | 13 | 24 | 16 | -1 | | |
| SV | 0 | | | | | | | | | |
| US | 0 | | | | | | | | | |
| WF | 0 | | | | | | | | | |

MOD_82CPR.CTRL

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| NW | 70 | 03 | | | | | | | | |
| WL | 1 | 16 | 18 | 12 | 18 | 12 | 18 | 12 | 18 | 12 |
| WL | 18 | 12 | 18 | 12 | 18 | 12 | 18 | 12 | 18 | 12 |
| WL | 18 | 12 | 11 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| WL | 1 | 7 | 12 | 18 | 12 | 18 | 12 | 18 | 12 | 18 |
| WL | 12 | 18 | 12 | 18 | 12 | 18 | 12 | 18 | 12 | 18 |
| WL | 12 | 18 | 12 | 8 | -1 | 0 | 0 | 0 | 0 | 0 |
| WL | 1 | 10 | 12 | 18 | 12 | 18 | 12 | 18 | 12 | 18 |
| WL | 12 | 18 | 12 | 18 | 12 | 18 | 12 | 18 | 12 | 18 |
| WL | 12 | 18 | 12 | 5 | -1 | 0 | 0 | 0 | 0 | 0 |
| SV | 1 | 0 | 24 | 1 | 48 | 1 | 0 | 0 | 0 | 0 |
| US | 1 | 23 | 25 | 47 | 49 | 0 | 0 | 0 | 0 | 0 |
| WF | 1 | | | | | | | | | |

long; the bits of word 23 will be in the most significant positions. This new 18-bit word will then be packed into a standard 32-bit word. The same procedure will be repeated for the 8 bits of word 47 and the 10 bits of word 49. The output data set will be in (13I9) format.

## 2.3 PUNS_MAY88.FOR

This program produces histograms and/or listings of the travel time ($\tau$) bursts within a specified range of sampling periods. This program was developed to give the user a first look at the distribution of the $\tau$ counts, within a single sampling period or for several sampling periods, before any further processing is done. Typically, the histograms are used to determine the acceptable range of 'good' $\tau$ counts to eliminate early and/or late echo returns from being used during the subsequent processing steps. The listings are used to establish the time base by determining the actual 'on bottom' sampling periods,

PUNS is applied to the data set produced by BUNS. The user specifies the types of output desired in a control file. As the BUNS data is read, each sampling period is counted consecutively. These 'record numbers' are used for specifying the samples which are to be plotted or listed.

Three types of histograms can be produced: (1) Level-1 (L1 option) produces one histogram for each sampling period within the range of record numbers specified by the user (START and END). (2) Level-2 (L2 option) produces a histogram for a group of sampling periods (GRPSIZ). Several records can be skipped (RATE) between subsequent groups to be plotted. These are repeated until all records between START and END have either been processed or skipped. (3) Level-3 produces a histogram of all processed records between START and END. This histogram is produced automatically every time the program is executed; that is, it is not a user-controlled option. To select either a level-1 or level-2 histogram, the user specifies 'L1' or 'L2' in the name list group called CARD8 in the control file. The bin sizes of the histograms are determined within the program from the range of $\tau$ counts specified by the user. Maximum and minimum counts (UBNDA and LBNDA, respectively) are supplied within the name list group CARD6 of the control file. A wide range can be selected to obtain a histogram of all $\tau$ counts or narrow one can be chosen to enlarge a portion of the count range. If the IES has two echo detectors, separate histograms are produced for the $\tau$'s from each detector. The user must specify the range of $\tau$ counts for both echo detectors for these histograms.

The listings of the travel times are either of integer counts ('IN' option) or their decimal equivalents ('DE' option). The user specifies either 'IN' or 'DE' within CARD8 of the control file to select the desired output. The decimal equivalents are calculated by scaling the integer counts by the factors SF1 and SF2 supplied by the user in CARD5 of the control file. If there are additional sensors, such as pressure, their values are given only as integer counts on both types of listings. The listings give the consecutive record number, sequence number, and the data values for each sampling period between START and END. A level-3 histogram will be produced for all records listed.

The FORTRAN source code is listed in Section 3.2. The user supplied name lists are listed in detail below.

## Control File

The control file is made up of eight name list groups, names CARD1 – CARD8. These are all in free format.

### CARD1

**HEADR** - (CHARACTER*60) Alphanumeric array containing comment information. Usually used to identify the instrument site and serial number.

### CARD2

**NTT** - (INTEGER*4) Number of travel time echo detectors on the IES.

**TTYPE(2)** - (CHARACTER*3) Alphanumeric names used to designate the types of echo detectors used.

### CARD3

**NWORDS** - (INTEGER*4) Number of words associated with each sampling period.

**LBURST** - (INTEGER*4) Number of $\tau$'s measured during a single sampling period.

**LBFST** - (INTEGER*4) Word number associated with the first $\tau$ of the burst. Typically, word 1 is the sequence number and the burst begins in word 2.

**RDFMT** - (INTEGER*4) Format of the input data. If 0, the data is binary. If 1, the data is in (13I9) format.

### CARD4

**NSEN** - (INTEGER*4) Number of sensors in addition to the $\tau$ echo detectors.

**SENSOR(3)** - (CHARACTER*2) Alphanumeric name for the type of sensor. 'PR' is for pressure, 'TP.' for temperature, and 'AM' for ambient noise.

**SWDNO(3)** - (INTEGER*4) Word number associated with the sensor.

### CARD5

**SF1** - (REAL*4) Scaling factor for the first $\tau$ echo detector used to convert $\tau$ from integer counts to time in decimal seconds.

**SF2** - (REAL*4) Same as above, except for the second $\tau$ echo detector. If there is only one $\tau$ detector, this variable is ignored.

## CARD6

**LBNDA** - (INTEGER*4) Lower limit of the histogram of counts for the first $\tau$ echo detector.

**UBNDA** - (INTEGER*4) Upper limit of the histogram of counts for the first $\tau$ echo detector.

**LBNDB** - (INTEGER*4) Same as LBNDA, except for the second echo detector. This variable and UBNDB are ignored if there is only one $\tau$ detector.

**UBNDB** - (INTEGER*4) Same as UBNDA, except for the second echo detector.

## CARD7

**START** - (INTEGER*4) Record number associated with the first sampling period to process. Counted sequentially from the beginning of the input data set.

**END** - (INTEGER*4) Record number associated with the last sample to process.

**RATE** - (INTEGER*4) Number of records to skip between the groups being processed. If RATE $> 0$, level–2 plots are generated.

**GRPSIZ** - (INTEGER*4) Number of records to be included in one histogram. It should always be greater than or equal to one.

**SEQINC** - (INTEGER*4) Expected increment of the sequence number between sampling periods. In the IES, this increase by 1 every 15 minutes. Thus for a 30 minute sampling period, SEQINC = 2.

## CARD8

**OPTN(4)** - (CHARACTER*2) Alphanumeric codes indicating the type of output desired. If no options are selected, only a level–3 histogram will be produced. Available options are:

'IN' - integer listing of the $\tau$ counts

'DE' - decimal listing of the $\tau$'s in seconds

'L1' - histogram for each sampling period

'L2' - histogram of groups of sampling periods

## 2.4 MEMOD_JUL89.FOR

The main objectives of MEMOD are to to establish the time base and convert the travel time counts to seconds. If the instrument is a PIES, MEMOD will also calibrate pressure and temperature. The inputs are the BUNS dataset and a control file. On output, a data file is created containing the calibrated measurements with their corresponding sample times. A listing file is also created; it contains statistical information pertaining to the travel time calculation.

The FORTRAN source code is given in Section 3.4. The user supplied control file is described below.

### 2.4.1 PROCESSING OF TRAVEL TIME

A single value is determined that suitably represents the burst of 'M' travel time measurements (typically M=24). First, the 'M' pings are windowed to remove unreliable $\tau$'s. Then the subroutine TTMODE calculates the modal $\tau$ based on the assumption that the $\tau$'s are members of a Rayliegh-distributed statistical population. Alternatively, the user may specify that the median $\tau$ of the burst be selected using the subroutine TTMEDN.

MEMOD is equipped to deal with IESs with one or two echo detectors. The measurements from one or both of the detectors may be processed in a single execution of MEMOD. The user specifies which method (median or mode) is to be used and with which detector (TT1 and/or TT2) within the control file.

To indicate to MEMOD that the travel time counter overranged, the window limits (in the control file) are set such that the value of the lower limit exceeds the upper limit. In that case, the upper limit and the measured $\tau$'s are recalculated by MEMOD by adding the appropriate power of two number of counts prior to windowing the $\tau$'s .

The user specifies upper and lower window limits in CARD7 of the control file. If all the $\tau$'s in the burst are outside the specified range (either all greater than the upper limit, or all less than the lower limit), the 'selected' $\tau$ is set equal to the limit exceeded. If the quartile range of the burst is too large, the 25th percentile $\tau$ (based on empirical evidence) is used instead of the median or modal $\tau$ . The range, $\tau$ , and number of $\tau$'s from the burst actually used in the selection process are written to the listing file.

Another windowing operation called 'Bin' windowing is applied within MEMOD_JUL89 at the start of the subroutine TTMODE. (The code can be modified to have bin windowing within the main code rather than in the subroutine.) The basic idea of bin windowing is that the direct surface reflections will be most probable, and that there is a time period within which all the true echos would be expected to occur. This method divides the 13-bit range of 8192 counts into 64 equal intervals (128 counts). The bin containing the most occurrences is likely to contain the single most-representative travel time of the burst. The bin window consists of this most abundant bin and its two closest neighbors and has a range of $3 \cdot 128 = 384$ counts. Since bursts measured by a "healthy" IES typically have ranges less than 200 counts, the desired signal will be contained within this bin window.

### 2.4.2 PROCESSING OF ADDITIONAL SENSORS

The subroutine TEMPRS within MEMOD converts temperature and pressure counts to physical units. This version of MEMOD does not process ambient noise measurements, which is another optional configuration for the IES.

Temperature counts are converted into °C by a linear expression. Two calibration methods are possible. One method uses an 'ideal' equation; the other, an empirical 'lab' equation. The choice of method is made in CARD14. With the present IESs, only the laboratory calibration should be used (specified with LAB=1 in CARD14). For this equation, the user supplies two calibration pairs (temperature and counts) in the namelist (NML) group CARD14.

The bottom pressure is a function of both the pressure counts and the temperature. The calibration equations used are specific to the Paroscientific Inc. sensors used. The calibration have two possible forms:

$$P = C\left[1-\left(\frac{T_0}{T}\right)^2 - D\left(1-\left(\frac{T_0}{T}\right)^2\right)^2\right] \qquad (1)$$

or
$$P = A\left(1-\frac{T_0}{T}\right) - B\left(1-\frac{T_0}{T}\right)^2 \quad (2)$$

The coefficients A, B, C, and the parameter $T_0$ are polynomial functions of temperature; D is a constant coefficient; and T is the measured period of the transducer (the counting period divided by the pressure counts). The user specifies which of the equations is to be used in CARD10 of the control file. Whenever possible, it is preferable to use Equation 1 instead of Equation 2.

The period, T, is determined from the pressure counts and the sampling interval. The user specifies, on CARD10 of the control file, whether or not the pressure counter has overranged at depth. If overranging has occurred, $2^{24}$ is added to the pressure counts prior to calculating the period. The user specifies the sampling interval length (in seconds) in CARD14 of the control file. If pressure has been electronically prescaled within the IES prior to recording, this sampling interval must be adjusted accordingly. Currently, the frequency output of the pressure sensor is divided by four before being counted, thus sampling interval specified should be divided by four.

The temperature-dependent coefficients (A, B, C, $T_0$) need to be recalculated for each sampling period. These coefficients have quadratic form ($T_0$ may occasionally be cubic), and they are unique for each transducer. Calibration coefficients are read from CARD11, CARD12 and CARD13.

### 2.4.3 TIME BASE

The exact day and time of a specific first ping of a burst serves as a reference from which all other sample times are determined. This time is specified in NML group CARD9. Typically the time of the first ping of the 'last-good-on-bottom' burst is used.

MEMOD introduces a small offset to the reference time specified in CARD9, so that it corresponds to the middle of the burst, rather than the first ping of the burst. (For a travel time burst consisting of 24 pings at 10 sec intervals, the time base is offset 115 sec.)

MEMOD and all further processing report time in units of yearhours; there are 8760 hours in a non-leap year. Zero yearhour corresponds to January 1 at 0000 UT. Thus positive yearhours correspond to sampling periods after January 1; negative yearhours refer to the previous calendar year.

### 2.4.4 OUTPUT DATA SET

On output, a data file and a listing file are created for each echo detector. The output data files consist of five variables written in 5E15.7 format. In order, these are travel time, pressure, temperature, ambient noise, and time (in units of seconds, decibars, °C , decibels, and yearhours). For IESs without the additional sensors, these variables contain only values of -99.00. The ambient noise column will always contain -99.00, since no processing is done on this variable.

## CONTROL FILE

The control file contains 9 NML groups, CARD1–CARD9, plus four additional groups for PIESs (CARD10–CARD14). All namelists are in free format.

**CARD1**

**HEADR** - (CHARACTER*60) string containing comment information. Usually used to identify the instrument site and serial number.

**CARD2**

**NTT** - (INTEGER*4) Number of echo detectors on the IES.

**TTYPE(2)** - (CHARACTER*3) strings used to designate the types of echo detectors used.

**CARD3**

**NWORDS** - (INTEGER*4) Number of words associated with each sampling period.

**LBURST** - (INTEGER*4) Number of $\tau$'s measured during a single sampling period.

**LBFST** - (INTEGER*4) Word number associated with the first $\tau$ of the burst. Typically, word 1 is the sequence number and the burst begins in word 2.

**RDFMT** - (INTEGER*4) Format of the input data. If 0, the data is binary. If 1, the data is in (13I9) format.

**CARD4**

**NSEN** - (INTEGER*4) Number of sensors in addition to the $\tau$ echo detectors. If 0, CARD10–CARD14 are not read by MEMOD.

**SENSOR(3)** - (CHARACTER*2) Character string name for the type of sensor. 'PR' is for pressure, 'TP' for temperature, and 'AM' for ambient noise.

**SWDNO(3)** - (INTEGER*4) Array containing the word number associated with the sensor type. SWDNO(i) indicates the word position of SENSOR(i).

**CARD5**

**SF1** - (REAL*4) Scaling factor for the first echo detector used to convert $\tau$ from integer counts to time in seconds.

**SF2** - (REAL*4) Same as above, except for the second echo detector. If NSEN = 1, this variable is not used.

**AMSF** - (REAL*4) Scaling factor used to convert the ambient noise counts to decibels. Currently, this variable is not used.

### CARD6

**NFIRST** - (INTEGER*4) Record number of the first sampling period to process. This is usually the first record containing 'on bottom' measurements.

**NFSEQ** - (INTEGER*4) Sequence number associated with the NFIRST record.

**NLAST** - (INTEGER*4) Record number of the last sampling period to process. This is usually the last record containing 'on bottom' measurements.

**NLSEQ** - (INTEGER*4) Sequence number associated with the NLAST record.

**SEQINC** - (INTEGER*4) Expected increment of the sequence number between sampling periods. In the IES, this increase by 1 every 15 minutes. Thus for a 30 minute sampling period, SEQINC = 2.

### CARD7

**LBND1** - (INTEGER*4) The lower bound on the $\tau$ counts for the first echo detector. Counts lower than LBND1 are excluded from further processing.

**UBND1** - (INTEGER*4) The upper bound on the $\tau$ counts for the first echo detector. Counts greater than UBND1 are excluded from further processing.

**LBND2** - (INTEGER*4) Same as LBND1, except for the second detector. Not used if NTT=1.

**UBND2** - (INTEGER*4) Same as UBND1, except for the second detector. Not used if NTT=1.

**DGRPHR** - (REAL*8) Number of sampling periods per hour.

### CARD8

**IOPT(6)** - (CHARACTER*4) string indicating the type of processing to be done. Available options are:

**'TT1'** - $\tau$ counts of the first echo detector are to be processed.

**'MED1'** - Subroutine TTMEDN is to be used to calculate the median of TT1 counts.

**'MOD1'** - Subroutine TTMODE is to be used to calculate the modal value of the TT1 counts.

**'TT2'** - Same as TT1, except for the second detector.

**'MED2'** - Same as MED1, except for TT2.

**'MOD2'** - Same as MOD1, except for TT2.

### CARD9

The first six of these variables specify the year, month, day, hour, minutes, and seconds to be associated with the sampling period whose sequence number is contained in, ISEQO. They are all supplied as two-digit numbers. The program assumes that it is the 20th century.

**IYR - (INTEGER*4)** year

**MNTH - (INTEGER*4)** month

**IDAY - (INTEGER*4)** day

**IHOUR - (INTEGER*4)** hour

**MINUT - (INTEGER*4)** minutes

**ISEC - (INTEGER*4)** seconds

**ISEQO - (INTEGER*4)** Sequence number of the sampling period which corresponds to the day and time specified by the preceeding six variables. This is used to establish the time base.

### CARD10

**EQN** - (CHARACTER*2) The equation to be used to calculate the pressure in dbar from the number of counts. The options are 'AB' or 'CD' corresponding to Equations 2 and 1.

**OVERNG** - (CHARACTER*2) Code to determine whether the pressure counts have overranged. Available codes are 'YE' – that overranging has occurred, and 'NO' – that it has not occurred.

### CARD11

**AC1** - (REAL*8) The constant in quadratic equation used to calculate the temperature-dependent calibration coefficient A (if EQN = 'AB') or C (if EQN = 'CD').

**AC2** - (REAL*8) Same as AC1, except it is the first order coefficient.

**AC3** - (REAL*8) Same as AC1, except it is the second order coefficient.

### CARD12

**BD1** - (REAL*8) The constant used to calculate the temperature-dependent calibration coefficient B (if EQN = 'AB') or D (if EQN = 'CD').

**BD2** - (REAL*8) Same as BD1, except it is the first order coefficient.

**BD3** - (REAL*8) Same as BD1, except it is the second order coefficient. If BD2 = BD3, then D will be a constant equal to BD1.

### CARD13

**T1** - (REAL*8) The constant used to calculate the temperature-dependent calibration coefficient $T_0$.

**T2** - (REAL*8) Same as T1, except it is the first order coefficient.

**T3** - (REAL*8) Same as T1, except it is the second order coefficient.

**T4** - (REAL*8) The third order coefficient, which is not used if BD2 = BD3.

### CARD14

**LAB** - (INTEGER*4) If LAB = 1, laboratory calibrations will be used to convert temperature counts to degrees centigrade. If LAB = 0, an idealized formula will be used. Only LAB=1 should be used.

**TSEC** - (REAL*4) Counting period in seconds for pressure. Typically, this is 1800 sec, however, if a SD PIES is in power-save mode the time would be shorter. If pressure has been electronically prescaled within the IES, this sampling interval must be adjusted accordingly.

**TREF1** - (REAL*4) First reference temperature of the laboratory calibrations.

**TREF2** - (REAL*4) Second reference temperature of the laboratory calibrations.

**CTREF1** - (INTEGER*4) Counts corresponding to TREF1.

**CTREF2** - (INTEGER*4) Counts corresponding to TREF2.

## EXAMPLE CONTROL FILES

Two examples of control files are listed in Table 3. In the first example, the IES has a single echo detector and pressure and temperature sensors. The input BUNS data are in binary format. The $\tau$ burst consists of 24 measurements, which are contained in words 2–25 of the data record. The pressure and temperature measurements are in word positions 26 and 27, respectively. The sequence number, held in word 1, will increment by 2. There will be two sampling periods in one hour, thus the sampling interval will be 30 minutes. Only the records from 58 to 17627 (with corresponding sequence numbers of 53 to 35191) will be processed. The time base is established by assigning the record with sequence number 35191 to the time of 00:59:27 UT on 17 January, 1985.

For each sample burst, the representative $\tau$ will be determined as the median value from the subset of all measurements with counts between 7280 and 7700. This median $\tau$ will be divided by 20480 Hz to convert it to seconds. The temperature counts are converted to °C using laboratory calibrations, where a temperature of 1°C corresponded to counts of 4554; and a temperature of 10°C , to 46260 counts. The pressure counts did not overrange, and the period of the oscillator will be determined by dividing the counts into 450.0 s (30 minute sampling interval divided by a prescaler of 4). Equation 2 will be used to determine the pressure (in psi, and this is scaled to decibars). CARDS11, 12, and 13 contain the coefficients A, B, and $T_0$.

In the second example, the IES has two echo detectors (types TTA and TTB) and no other sensors. The format of the BUNS data is 13I9. Each sample burst consists of 32 pings; since both detectors receive the return echoes, there are 64 $\tau$ measurements for each sampling period. These measurements are stored in words 2-65 of the data record. For the first echo detector, the $\tau$'s within the limits 99650 and 100325 will be used to determine a single $\tau$ by the mode method. The $\tau$'s from the second detector that pass through the 1545–1840 window will be used calculate the $\tau$ by the median method. In both cases, the calculated $\tau$ is scaled by 20480.0 Hz. The time base is established by assigning the sampling period with sequence number 707 to 11:45:00 UT on 16 July 1982. There are four sampling periods per hour, thus the sampling interval is 15 minutes and the sequence number will increment by 1. Since there are no additional sensors, CARDS10–CARD14 are not required.

Table 3: Two examples of control files for MEMOD_JUL89.FOR. See text for explanation.

Control File 1

```
$CARD1
       HEADR='Example 1:  IES with pressure and temperature'
$END
$CARD2  NTT=1, TTYPE='TTB', '    ' $END
$CARD3  NWORDS=27, LBURST=24, LBFST=2, RDFMT=0 $END
$CARD4  NSEN=2, SENSOR='PR', 'TP',' ', SWDNO=26, 27,0 $END
$CARD5  SF1=20480.0, SF2=0.0, AMSF=0.0 $END
$CARD6  NFIRST=58, NFSEQ=53, NLAST=17627, NLSEQ=35191, SEQINC=2 $END
$CARD7  LBND1=7280, UBND1=7700, LBND2=0, UBND=20, DGRPHR=2.00D+00 $END
$CARD8  IOPT=' TT1', 'MED1',2*'    ' $END
$CARD9  IYR=85, MNTH=01, IDAY=17, IHOUR=00, IMIN=59, ISEC=27, ISEQ0=35191 $END
$CARD10 EQN='AB', OVERNG='NO' $END
$CARD11 AC1=5.18004E+04, AC2=-9.70308E-01, AC3=1.71739E-03
$CARD12 BD1=3.17505E-05, BD2=-7.80773E-01, BD3=1.04970E-02
$CARD13 T1=2.597996E-05, T2=-1.99543E-11, T3=1.70393E-13,T4=0 $END
$CARD14 LAB=1, TSEC=450.0, TREF1=1.0, TREF2=10.0, CREF1=4554, CREF2=46260  $END
```

Control File 2

```
$CARD1
       HEADR='Example 2: IES with two travel time echo detectors'
$END
$CARD2  NTT=2, TTYPE=' TTA', ' TTB' $END
$CARD3  NWORDS=65, LBURST=32, LBSFT=2, RDFMT=1 $END
$CARD4  NSEN=0, SENSOR=3*'  ', NWORD=3*0 $END
$CARD5  SF1=20480.0, SF2=20480.0, AMSF=0.0 $END
$CARD6  NFIRST=78, NFSEQ=75, NLAST=710, NLSEQ=707, SEQINC=1 $END
$CARD7  LBND1=99650, UBND1=100325, LBND2=1545, UBND2=1840, DGRPHR=4.00D+00 $END
$CARD8  IOPT=' TT1', 'MOD1', ' TT2', 'MED1' $END
$CARD9  IYR=82, IMNTH=07, IDAY=16, IHOUR=11, IMIN=45, ISEC=00, ISEQ0=707 $END
```

## 2.5 FILL_JAN91.FOR

FILL checks the data set for proper incrementing of the time base and corrects the errors encountered. Two types of time base errors can occur: 1) a complete record from a sampling period can be missing or 2) the time associated with a sampling period can be incorrect.

FILL steps through the MEMOD output, checking that the time increment between successive samples equals the expected value of DELTAT, specified by the user in the control file. If errors are found, the 'out-of-sequence' records are saved in arrays. When proper incrementing resumes, FILL checks the records stored in the arrays for the two types of errors listed above. If a record has an incorrect time associated with the measurements, only the time is corrected and the data values are not adjusted. However if a complete sampling period is missing, the gap is filled with data values which have been interpolated between neighboring good records, and the correct time is associated with these values. All records which require a correction are counted (or 'flagged').

When the two types of errors are intermingled, that is samples are missing within a period which has incorrect times, then missing samples are inserted before the group of samples with incorrect times. If isolated good records are interspersed in such a section, missing records will be added so as to preserve the good records' true positions.

The output consists of both a log file and a corrected data file. If the instrument is a PIES, two additional data files are created: one for pressure and one for temperature. The individual data files will contain the proper time base associated with that particular sensor type and PIES model (URI or Sea Data). The log file lists the records which were out-of-sequence and how many additional records were needed to fill any gaps. The total number of flagged records are also reported. The output data files contain two variables in 2E15.7 format with time in the second column.

The FORTRAN source code is listed in Section 3.3. The user supplied control parameters are given below.

### CONTROL FILE

The control file is composed of three NML groups, CARD1–CARD3. These are in free format.

**CARD1**

**HEADR** - (CHARACTER*60) A string containing comment information. Usually used to identify the instrument site and serial number.

**CARD2**

**NSTART** - (INTEGER*4) Sequential number of first record to start checking the times. All records prior to this one are assumed to be in correct order and are written to the output data set without being checked.

**NSTOP** - (INTEGER*4) Sequential number of last record to check for incorrect timing. All subsequent records are written to the output data set without being checked.

**MAXDLT** - (INTEGER*4) Maximum allowable time gap in hours. If this limit is exceeded, execution of the program terminates.

**DELTAT** - (INTEGER*4) Sampling interval in hours.

## CARD3

**PRESS** -(CHARACTER*3) The answer to whether the instrument is a PIES or not. Only the first character is checked; 'Y' or 'y' indicates a PIES, and a pressure file and a temperature file are additionally created.

**MODEL** -(CHARACTER*3) Is the PIES a URI or Sea Data (SD) model? Only the first character is checked; 'S' or 's' indicates a SD instrument. This information is used to determine which set of offsets to apply to $\tau$ time base to get the appropriate time bases for pressure and for temperature.

## 2.6 DETIDE_AUG90.FOR

DETIDE reduces the effect of the tide in the travel time record (Fig. 5). The tidal signature in

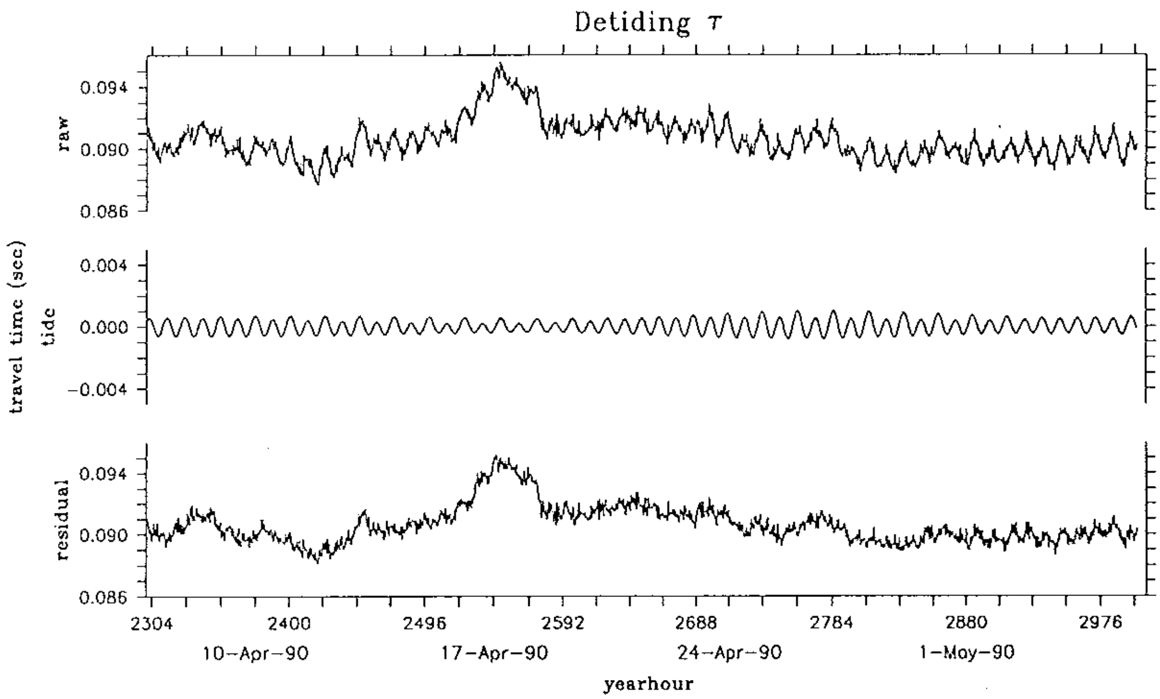



Figure 5: The measured travel time, the prediction of the tide's effect on travel time, and the 'detided' travel time.

travel time is composed of two opposing effects. For a tidal elevation $\eta$, the acoustic path increases by $2\eta$, but the speed of sound increases due to the increase in hydrostatic pressure ($\rho g \eta$). The two effects may be expressed as

$$\Delta\tau_{path}(\eta) = \frac{2\eta}{c_s}$$

$$\Delta\tau_{press}(\eta) = \frac{2H}{c_s+\Delta c_s(\eta)} - \frac{2H}{c_s} = \frac{-2H\frac{\Delta c_s(\eta)}{c_s}}{\left(1+\frac{\Delta c_s(\eta)}{c_s}\right)} \approx -2H\frac{\Delta c_s(\eta)}{c_s}$$

here $c_s$ is the speed of sound for the region of mean depth $H$; $\delta c_s$ is the variation in sound speed resulting from the tidal height, $\eta$. These expressions may be combined and simplified by using binomial expansion and by utilizing the approximately linear relation between sound speed and pressure. The net change in travel time can be expressed as

$$\Delta\tau = \frac{2\eta}{c_s\gamma}, \qquad \text{where} \qquad \gamma = 1 + H\left(\frac{\rho g}{c_s}\frac{\partial c_s}{\partial p}\right)$$

The user supplies $c_s$ and $\gamma$ (CBAR and PFACTR) in the control file. CBAR is determined for the region and depth from the Matthews table (found in Handbook of Oceanographic Tables, Section III, Table 11. Bialek, 1966). The PFACTR may be calculated from the instrument depth ($\frac{\rho g}{c_t}\frac{\partial c_t}{\partial p} \approx 1.1 \times 10^{-5} s^{-1}$). The scaled tidal heights are subtracted from the measured $\tau$'s to create a set of 'detided' $\tau$'s.

The tide ($\eta$) is estimated using the amplitude and phases, $H$ and $g$, of eight of the most significant tidal constituents ( $M_2$, $N_2$, $S_2$, $K_2$, $K_1$, $O_1$, $P_1$, and $Q_1$) which are supplied in the control file[2].

The tidal signal is predicted by

$$\eta(t) = \sum_{n=1}^{8} H_n f_n \cos(\omega_n t - g_n + V_n + u_n)$$

where $\omega_n$ is the angular frequency of the n'th constituent, $V_n$ is the phase of the equilibrium tide at time zero, $f_n$ is the nodal factor, $u_n$ is the nodal correction. The time, $t$, associated with each sampling period is referenced to Universal Time.

The control file contains the time-dependent node factor ($f$) and equilibrium argument ($V_o + u$) for each constituents. The $f_n$ and $u_n$ factors account for small but significant variations resulting from the modulation of both $H_n$ and $g_n$ with the regression of the moon's ascending node. The factors $f$ and $u$ are considered constant for any one year, but varies from year to year with the regression's period of 18.6 years. (For solar constituents, $f = 1$ and $u = 0$.) The yearly values are tabulated in the literature pertaining to tidal prediction by Harmonic Analysis (e.g., Tables 14 and 15 in Schureman, 1941). Tables 4 and 5 list the nodal factors and the equilibrium arguments, of the eight constituents for years 1973 to 1999.

Since the tides are generated sequentially, the input data set must not be missing any records; thus the data set produced by FILL is used as the input.

The output consists of a log file and a disk data file. The log file lists the tidal components used to generate the tidal amplitudes. The disk data set consists of seven variables written in (4E15.7) format. In order, these are: measured $\tau$ (in seconds), detided $\tau$ (in seconds), predicted tide (in seconds), and time (in yearhours).

The FORTRAN source code is listed in Section 3.5. The user supplied control are listed below.

## CONTROL FILE

The control file consists of seven name list groupings, CARD1 – CARD7, which are in free format.

### CARD1

**HEADR** - (CHARACTER*60) Alphanumeric array containing comment information.

[2]The amplitudes and phases were derived using the results of Response Analysis applied to the bottom pressure records. Response analysis is described in Section 2.9. The bottom-pressure tidal signal is related to the sea surface elevation by the hydrostatic equation.

Table 4: Node Factor, $f$, for middle of each calendar year, 1973 to 1999 (Schureman, 1941)

| Year | Contituent | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $M_2$ | $N_2$ | $S_2$ | $K_2$ | $K_1$ | $O_1$ | $P_1$ | $Q_1$ |
| 1973 | 0.995 | 0.995 | 1.000 | 1.055 | 1.029 | 1.047 | 1.000 | 1.047 |
| 1974 | 1.008 | 1.008 | 1.000 | 0.957 | 0.991 | 0.984 | 1.000 | 0.984 |
| 1975 | 1.020 | 1.020 | 1.000 | 0.871 | 0.951 | 0.920 | 1.000 | 0.920 |
| 1976 | 1.029 | 1.029 | 1.000 | 0.804 | 0.916 | 0.863 | 1.000 | 0.863 |
| 1977 | 1.035 | 1.035 | 1.000 | 0.763 | 0.891 | 0.822 | 1.000 | 0.822 |
| 1978 | 1.038 | 1.038 | 1.000 | 0.748 | 0.882 | 0.806 | 1.000 | 0.806 |
| 1979 | 1.036 | 1.036 | 1.000 | 0.760 | 0.890 | 0.819 | 1.000 | 0.819 |
| 1980 | 1.030 | 1.030 | 1.000 | 0.799 | 0.913 | 0.858 | 1.000 | 0.858 |
| 1981 | 1.021 | 1.021 | 1.000 | 0.864 | 0.948 | 0.915 | 1.000 | 0.915 |
| 1982 | 1.009 | 1.009 | 1.000 | 0.949 | 0.987 | 0.979 | 1.000 | 0.979 |
| 1983 | 0.997 | 0.997 | 1.000 | 1.045 | 1.026 | 1.041 | 1.000 | 1.041 |
| 1984 | 0.984 | 0.984 | 1.000 | 1.142 | 1.060 | 1.096 | 1.000 | 1.096 |
| 1985 | 0.974 | 0.974 | 1.000 | 1.226 | 1.086 | 1.140 | 1.000 | 1.140 |
| 1986 | 0.967 | 0.967 | 1.000 | 1.285 | 1.104 | 1.168 | 1.000 | 1.168 |
| 1987 | 0.964 | 0.964 | 1.000 | 1.315 | 1.112 | 1.182 | 1.000 | 1.182 |
| 1988 | 0.964 | 0.964 | 1.000 | 1.310 | 1.111 | 1.180 | 1.000 | 1.180 |
| 1989 | 0.969 | 0.969 | 1.000 | 1.270 | 1.100 | 1.161 | 1.000 | 1.161 |
| 1990 | 0.977 | 0.977 | 1.000 | 1.203 | 1.079 | 1.128 | 1.000 | 1.128 |
| 1991 | 0.998 | 0.998 | 1.000 | 1.115 | 1.051 | 1.081 | 1.000 | 1.081 |
| 1992 | 1.000 | 1.000 | 1.000 | 1.016 | 1.015 | 1.024 | 1.000 | 1.024 |
| 1993 | 1.013 | 1.031 | 1.000 | 0.922 | 0.976 | 0.960 | 1.000 | 0.960 |
| 1994 | 1.024 | 1.024 | 1.000 | 0.842 | 0.937 | 0.897 | 1.000 | 0.897 |
| 1995 | 1.032 | 1.032 | 1.000 | 0.785 | 0.905 | 0.844 | 1.000 | 0.844 |
| 1996 | 1.037 | 1.037 | 1.000 | 0.754 | 0.886 | 0.812 | 1.000 | 0.812 |
| 1997 | 1.038 | 1.038 | 1.000 | 0.750 | 0.883 | 0.808 | 1.000 | 0.808 |
| 1998 | 1.034 | 1.034 | 1.000 | 0.772 | 0.897 | 0.832 | 1.000 | 0.832 |
| 1999 | 1.027 | 1.027 | 1.000 | 0.821 | 0.926 | 0.879 | 1.000 | 0.879 |

Table 5: Equilibrium Argument ($V_o + u$) for the Greenwich Meridian at the beginning of each calendar year, 1973 to 1999 (Schureman, 1941)

| Year | Contituent | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $M_2$ | $N_2$ | $S_2$ | $K_2$ | $K_1$ | $O_1$ | $P_1$ | $Q_1$ |
| 1973 | 84.5 | 270.0 | 0.0 | 218.4 | 19.0 | 61.7 | 349.5 | 247.2 |
| 1974 | 185.3 | 282.0 | 0.0 | 218.2 | 19.2 | 162.0 | 349.7 | 258.7 |
| 1975 | 285.8 | 293.8 | 0.0 | 215.8 | 18.2 | 263.5 | 350.0 | 271.5 |
| 1976 | 26.0 | 305.3 | 0.0 | 211.5 | 16.2 | 6.5 | 350.2 | 285.8 |
| 1977 | 101.8 | 279.3 | 0.0 | 207.6 | 14.1 | 85.7 | 349.5 | 263.2 |
| 1978 | 201.8 | 290.6 | 0.0 | 200.9 | 10.5 | 191.3 | 349.7 | 280.1 |
| 1979 | 301.9 | 301.9 | 0.0 | 194.2 | 6.8 | 296.9 | 349.9 | 297.0 |
| 1980 | 42.0 | 313.3 | 0.0 | 188.2 | 3.7 | 41.6 | 350.2 | 312.9 |
| 1981 | 117.9 | 287.4 | 0.0 | 185.6 | 2.5 | 119.4 | 349.4 | 289.0 |
| 1982 | 218.4 | 299.2 | 0.0 | 183.1 | 1.5 | 221.1 | 349.4 | 301.9 |
| 1983 | 319.1 | 311.2 | 0.0 | 182.6 | 1.5 | 321.4 | 349.9 | 313.5 |
| 1984 | 60. | 323.4 | 0.0 | 184.2 | 2.4 | 60.8 | 350.2 | 324.2 |
| 1985 | 136.8 | 298.4 | 0.0 | 189.3 | 4.9 | 134.2 | 349.4 | 295.8 |
| 1986 | 238.1 | 311.0 | 0.0 | 193.5 | 6.9 | 232.6 | 349.6 | 305.4 |
| 1987 | 339.6 | 323.7 | 0.0 | 198.3 | 9.2 | 330.7 | 349.9 | 314.8 |
| 1988 | 81. | 336.4 | 0.0 | 203.3 | 11.6 | 68.8 | 350.1 | 324.2 |
| 1989 | 158.1 | 311.7 | 0.0 | 210.0 | 14.8 | 141.6 | 349.4 | 295.3 |
| 1990 | 259.4 | 324.3 | 0.0 | 213.9 | 16.7 | 240.1 | 349.6 | 305.0 |
| 1991 | 0.5 | 336.7 | 0.0 | 216.6 | 18.0 | 339.0 | 349.8 | 315.2 |
| 1992 | 101.3 | 348.8 | 0.0 | 217.6 | 18.7 | 78.7 | 350.1 | 326.1 |
| 1993 | 177.6 | 323.3 | 0.0 | 218.5 | 19.4 | 154.0 | 349.3 | 299.7 |
| 1994 | 278.0 | 334.9 | 0.0 | 215.4 | 18.0 | 256.1 | 349.6 | 313.0 |
| 1995 | 18.2 | 346.4 | 0.0 | 210.3 | 15.6 | 359.7 | 349.8 | 327.9 |
| 1996 | 118.3 | 357.8 | 0.0 | 204.0 | 12.2 | 104.8 | 350.1 | 344.3 |
| 1997 | 194.0 | 331.7 | 0.0 | 199.2 | 9.5 | 185.2 | 349.3 | 322.9 |
| 1998 | 294.0 | 83.0 | 0.0 | 192.7 | 6.0 | 290.5 | 349.6 | 339.5 |
| 1999 | 34.2 | 354.5 | 0.0 | 187.2 | 3.2 | 34.6 | 349.8 | 354.9 |

## CARD2

**NFIRST** - (INTEGER*4) The number of the first record to process.

**NLAST** - (INTEGER*4) The number of the last record to process.

**IYR** - (INTEGER*4) Year for which the tidal parameters are to be calculated.

**DELT** - (REAL*4) Sampling interval in hours.

## CARD3

**H(8)** - (REAL*4) Array of half amplitudes in centimeters for tidal constituents in the following order: $M_2$, $N_2$, $S_2$, $K_2$, $K_1$, $O_1$, $P_1$, and $Q_1$.

## CARD4

**PHI(8)** - (REAL*4) Array of phases in degrees (Greenwich epoch) corresponding to the amplitudes given in H.

## CARD5

**F(8)** - (REAL*4) Array of f node parameters for the middle of the calendar year. Given in the same order as above.

## CARD6

**VU(8)** - (REAL*4) The equilibrium argument ($V_o$ + u) for the Greenwich meridian at the beginning of the calendar year. Given in the same order as above.

## CARD7

**CBAR** - (REAL*4) The average sound velocity for the location and depth of the IES.

**PFACTR** - (REAL*4) Factor used to modify the speed of sound (CBAR) to account for the variation in sound speed resulting from the tidal variation in pressure ($\rho g \eta(t)$).

**LOCN** - (CHARACTER*10) The string used to identify the location and depth to which CBAR and PFACTR apply.

## 2.7 DESPIKE_AUG90.FOR and DESPIKE_TP.FOR

DESPIKE identifies spikes in the measurements and replaces them with interpolated values. A spike is defined either as a measurement which exceeds specified limits or as one which increases (or decreases), from the preceeding few measurements, more rapidly than a specified rate. Figure 1 illustrates a travel time record, both before and after being processed by despike.

There are two DESPIKE programs: DESPIKE_TP.FOR which is applied to temperature or pressure files (from the FILL step), and DESPIKE_AUG90.FOR which runs on the travel time (from DETIDE). The difference in the programs only amounts to operating on different columns of the file.

Within the control file, the user specifies the upper and lower bounds within which the measurements are considered reliable (VMAX and VMIN). Values outside of the specified range are replaced. The maximum gradient (SLOPE1) to be tolerated is specified in CARD3.

Upon execution, DESPIKE first checks whether a measurement falls within the specified range. If the measurement meets this criterion, the value is then tested to make sure that it has not changed by an amount that is larger than expected relative to the average of the previous LAVG1 points ('slope method'). A measurement which fails either of these two tests is stored in a temporary array. The subsequent measurements are tested and, if they also fail either test, are stored in the array. When the next 'good' (one which meets both criteria) value is found, all spikes stored in the array are replaced with values which have been interpolated between the neighboring good values. For travel time, this procedure is applied to the detided $\tau$. The measured $\tau$ is adjusted accordingly by adding the appropriate tidal height (in seconds).

The initial running-average, to initiate the 'slope method' of despiking (RNAVG1) is specified within CARD3 of the control file. A RNAVG1 value of zero directs DESPIKE to simply use the average of the first LAVG1 points (Best used only when confident that the first LAVG1 points are free of spikes). Once started, the running average is updated each time a good data point is written to the output file.

The 'slope method' may be visualized as a beam emanating forward from a point at the center of an averaging interval. The width of the beam is controlled by SLOPE1. The width is also affected by the number of points in the running average, LAVG1. Consider a sample at a specific time being tested (good or spike). The distance between the test point and the origin of the beam increases with the length of the averaging interval; thus, for a given SLOPE1 increasing the LAVG1 widens the beam at the test point. However, LAVG1 also tunes the smoothness of the beam's path. As the beam is stepped forward in time, the path the beam follows is smoothed by this effective running-average filter.

The program 'flags' certain replacements for which it has some doubt. When a point fails the slope test but is less than the next consecutive 'good' point it is replaced by an interpolated value, but a flag is issued in the output list file.

Three output files are generated: 1) a list file which gives details of the replacements, 2) a log file that documents the control file used with a summary of the list file, and 3) the output data file, which has the same format as the DETIDE output data file, or the same format as the FILL data file if the variable is pressure or temperature.

The FORTRAN source code is listed in Section 3.6 and the contents of the control file are listed below.

## CONTROL FILE

Three NML groups make up this file, CARD1 – CARD3. They are all read in free format.

**CARD1**

**HEADR** - (CHARACTER*60) string containing comment information.

**CARD2**

**TINTVL** - (REAL*4) Time interval in hours between data points.

**VMAX** - (REAL*4) Upper bound delimiting acceptable measurements.

**VMIN** - (REAL*4) Lower bound delimiting acceptable measurements.

**CARD3**

**SLOPE1** - (REAL*4) The allowed rate of change per hour (in the same units as the data – seconds, dbars, or degrees Celsius).

**RNAVG1** - (REAL*4) Initial value for the running average of LAVG1 samples. If RNAVG1 = 0.0, the program computes its value by averaging the first LAVG1 points.

**LAVG1** - (REAL*4) Number of points used to compute the running average.

## 2.8 SEACOR_AUG90.FOR

SEACOR removes the effect of seasonal warming and cooling of the surface layers from the travel-time measurements. A long-term-average seasonal cycle is used to estimate this correction. For instance, in the Gulf Stream, the travel time varies seasonally 1–1.8 msec independent of lateral shifts in the Gulf Stream's position; this seasonal change would correspond to a 20–36 m bias error in the main thermocline depth, if not removed. The user supplies a seasonal correction curve for the

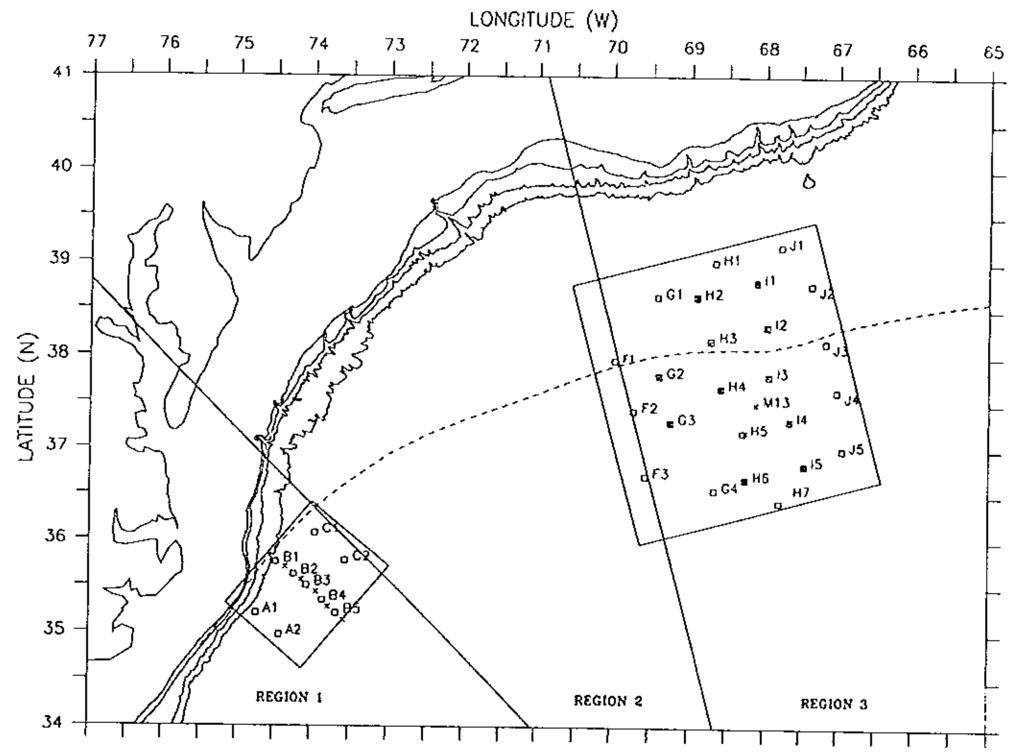



Figure 6: The three regions represent the dominant spatial variation of the seasonal corrections to travel-time.

specific oceanic region where the IES was deployed. This curve consists of 24 values, one for each month for a two-year period. The yearhours corresponding to these correction factors (assumed to be the first day of each month) are also specified. Currently, SEACOR has three sets of correction factors; these are initialized in DATA statements. The correction factors represent three different regions in the Gulf Stream. The particular set to be used is specified in the third NML group of the control file. For locations other than those shown in Figure 6 the SEACOR code must be modified to recognize a new region specification. Figure 7 shows the seasonal cycle for the three Gulf Stream regions.

The correction factors were determined with data from historical archives. Over 5000 XBT and CTD casts in the Gulf Stream region were examined for seasonal and regional variations. It was found that down-stream variation dominated the spatial dependence, and the three regions in

Figure 6 were chosen to represent the effect. The cross-stream dependence of the seasonal correction was considered insignificant in this region.

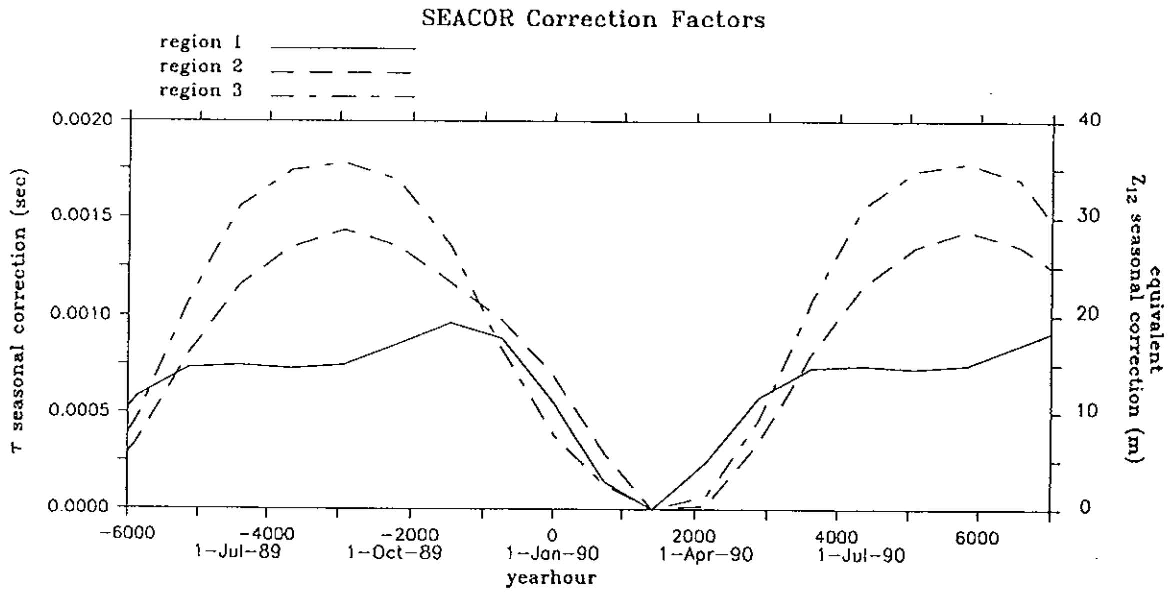



Figure 7: The seasonal correction factors for the three regions displayed in Figure 6.

The data set produced by DESPIKE is used as input. The correction factor to be used for each sampling period is determined by linearly interpolating between the monthly values stored in the array. Then this correction factor is added to both the measured and detided $\tau$'s of each sampling period.

Within the control file, the user specifies if the deployment period spans one or two calendar years and if any year involved is a leap year. The appropriate yearhours, associated with the monthly correction factors, are adjusted when either of these years is a leap year.

The output consists of two files: A log file, which lists the monthly correction factors and their associated yearhours; and the data file, which contains the seasonally corrected $\tau$'s in the same format as the DESPIKE output data file.

The FORTRAN source code is listed in Section 3.7.

## CONTROL FILE

This file is composed of three NML groups, CARD1 – CARD3.

### CARD1

**HEADR** - (CHARACTER*80) Alphanumeric variable containing comment information.

### CARD2

**NPTS** - (INTEGER*4) Number of sampling periods to process.

**NOYRS** - (INTEGER*4) Number of calendar years spanned by the dataset.

**FRSTYR** - (CHARACTER*2) Alphanumeric code designating whether or not the first year is a leap year. Options are 'YE' or 'NO'.

**SCNDYR** - (CHARACTER*2) Same as FRSTYR, except for the second year.

### CARD3

**REGION** - (I1) A number specifying the geographic region of the record.

## 2.9 RESPO_JUL88.FOR

RESPO removes the tide from the bottom pressure using Response Analysis to predict the tidal signal. Figure 8 illustrates detiding by RESPO in the time domain, while Figure 9 represents the frequency domain expression.

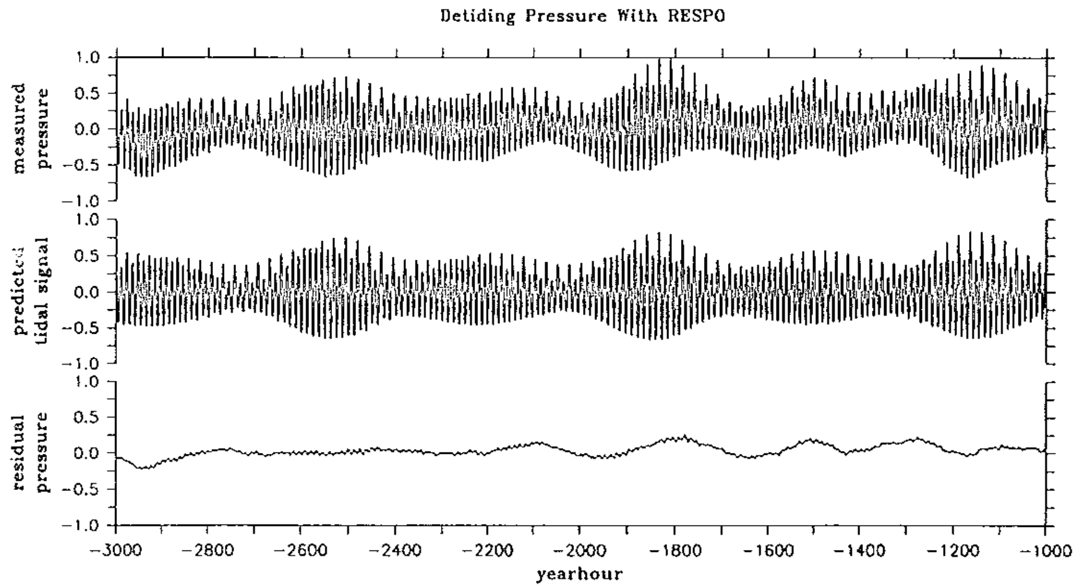



Figure 8: The uppermost panel displays the measured bottom pressure; the middle panel represents the portion of the measured bottom pressure resulting from the tide; and the bottom panel is the residual bottom pressure.

Response analysis constructs and applies a predictive filter which represents the ocean's response to gravitational forcing. Sometimes moderately nonlinear interactions and non-gravitational forcing (*e.g.*, the radiational tide) are included. Unlike the related harmonic analysis, the response analysis assumes nothing about which frequencies are present, because the input function is derived directly from the Newtonian-Keplerian orbital motions; the input function contains all the variations of the astronomic forcing regardless of size. The oceanic response is considered distinctly from the astronomic forcing. The method also has a more physical basis than harmonic analysis since it treats the ocean as a dynamical system.

A simple filter may be expressed as,

$$y(t) = \sum_k \ell(\tau_k) x(t - \tau_k), \tag{3}$$

where $y$ is the predicted tide, $x$ is the input function, and $\ell$ is the response of the ocean to a unit

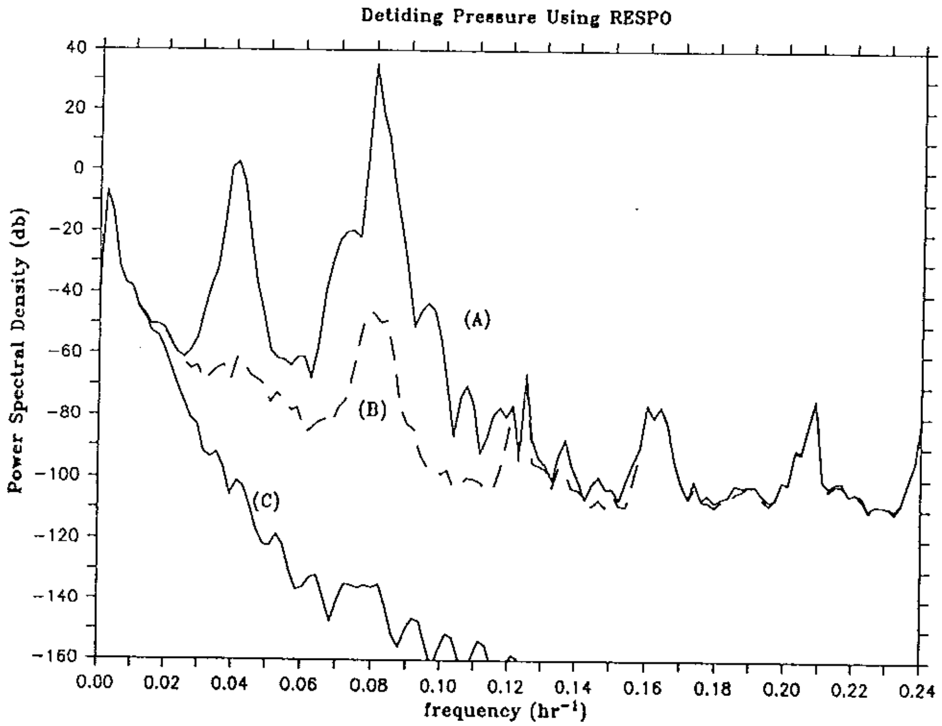



Figure 9: The power spectral density of the (A) measured bottom pressure, (B) the residual bottom pressure, and (C) the 40 hr low-pass filtered residual.

impulse of $x$ at time zero.

The tidal prediction illustrated by Equation 3 depends only on the input's temporal variation at that particular location. As a refinement the forcing at other locations may be included in the prediction:

$$y(t) = \sum_i \sum_k \ell_i(\tau_k) x_i(t - \tau_k), \tag{4}$$

where 'i' represents forcing at neighboring locations that might influence sea level at the site of interest. Response analysis systematically includes spatial dependence by expanding $x$ in surface spherical harmonics. The predicted tide is expressed as a filter acting on the complex-valued, time-varying amplitudes of the spherical harmonic functions representing the equilibrium-tidal potential.

$$y(t) = Re \sum_k \sum_{n=2}^{\infty} \sum_{m=0}^{n} w_n^m(\tau_k) C_n^m(t - \tau_k) \tag{5}$$

The indices n and m are the degree and order of the surface spherical harmonic functions. $C_n^m(t)$, which replaces $x$, is the set of time-varying amplitudes of the corresponding spherical harmonic functions. The $w_m^n$, which replace $\ell$, are the complex weights associated with each $C_n^m(t)$. Using the data to be detided, $(\eta(t_j))$, the weights $(w_m^n)$ are found by solving the overdetermined set of equations such that the difference between the data and the predicted tide $(\eta(t_j) - y(t_j))$ is minimized in a least-squares sense.

The equilibrium-tidal potential for a mass $M$, whose center of mass is at distance $\rho$ from the point of observation is:

$$\frac{V(t)}{g} = \frac{GM}{g\rho}$$

where $V(t)$ is the gravitational potential due to mass $M$, $G$ is the gravitational constant, and $g$ is local gravity. Typically, ninety-nine percent of the gravitational tidal variance can be explained with the equilibrium-tidal potential (due to the masses of the moon and sun) represented by just the $C_2^1$and $C_2^2$ $\{n$=2, $m$=1, 2$\}$ amplitude functions.

The spherical harmonics corresponding to $C_2^1$ and $C_2^2$ are illustrated as viewed down the axis of rotation in Figure 10. The plus sign represents bulging relative to the geoid, the minus sign, flattening. From this illustration, it is apparent that $C_2^1$ and $C_2^2$ are associated with the diurnal and semi-diurnal species of the harmonic analysis. RESPO is set up to use these functions ($C_2^1$ and $C_2^2$)

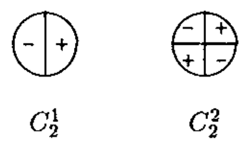



Figure 10: The spherical harmonics associated with $C_2^1$ and $C_2^2$ are viewed from the axis of rotation. Plus and minus signs correspond to bulging and flattening relative to the geoid.

as input at time-lags of $\tau_k = k * 48\ hr,\ k = -1, 0, 1$. Thus Equation 5 is then truncated to:

$$y(t) = Re \sum_{k=-1}^{1} \sum_{m=1}^{2} w_2^m(k*48) C_2^m(t - k*48) \tag{6}$$

The 6 weights, $w_2^m(k*48hr), m = 1,2\ k = -1,0,1$ are found from the overdetermined set of equations.

$$\eta(t_j) \quad = \quad Re\,(C(t_j) * W)\,, \tag{7}$$

where $\quad C(t_j) \quad = \quad [C_2^1(t_j+48)\ C_2^1(t_j)\ C_2^1(t_j-48)\ C_2^2(t_j+48)\ C_2^2(t_j)\ C_2^2(t_j-48)],$

and $\quad W \quad = \quad \begin{pmatrix} w_2^1(48) \\ w_2^1(0) \\ w_2^1(-48) \\ w_2^2(48) \\ w_2^2(0) \\ w_2^2(-48) \end{pmatrix}$

These weights are applied to the $C_2^m$ to generate the predicted tide. This is subtracted from the original data $\eta(t_j)$ to give the residual tide.

## Use

RESPO operates on the DESPIKE output file and a control file. It creates an output data file and a log file. The four columns of the output data file contain the raw pressure, detided pressure, predicted tide, and sample time. The log file contains relevant information about the response analysis and the equivalent harmonic constituents.

## Control File

The control file consists of 3 namelist groups. They are assigned the names CARD1, CARD2, and CARD3. A sample control file has the form shown below.

```
$CARD1
    HEADR= 'pies89g2_213    RESPO ' $end
$CARD2
    FORM= '(45X,E15.7,30X,E15.7)' $end
$CARD3 length=22772, year=1989, yearhr=-5253.253125, d=0.500004845 $end
```

The namelists are read free format. The variables' data types and definitions are listed below:

### CARD1

**HEADR** -(CHARACTER*40) This is the Header to be used in the output log file. HEADR typically contains the site and recovery cruise number in order to identify the record. The string should be enclosed in quotes.

### CARD2

**FORM** -(CHARCTER*40) This string is format specification for reading pressure and time, in that order. In the case of a non-standard input file, in which time comes before pressure the "TL" format specifier may be used to space backwards, after reading pressure, to read time. For a 7E15.7 file, the format '(90X,E15.7,TL60,E15.7)' would read the 7th entry in the record and then the 4th.

### CARD3

**LENGTH** -(INTEGER*4) The number of sample records in the time series.

**YEAR** -(INTEGER*4) Reference year from which time in yearhour is expressed. For example, if 1-Jan-1990 is defined as yearhour 0000Z then YEAR=1990.

**YEARHR** -(REAL*8) Yearhour of first pressure sample. This time corresponds to the center of the half-hour measurement period.

**D** -(REAL*4) The interval between successive samples. Nominally D is 0.5 hr; slight drifts over long deployments may lead to an effective sampling interval differing from this by a part in one million, which is accounted for in double precision.

## 2.10 FILTER_NAMES.M

Residual pressure, temperature, and travel time are filtered and subsampled using functions from MATLAB and MATLAB's Signal Processing Toolbox. The functions were collected into a routine called FILTER_NAMES.M. Additionally, travel time ($\tau$) is scaled to $Z_{12}$.

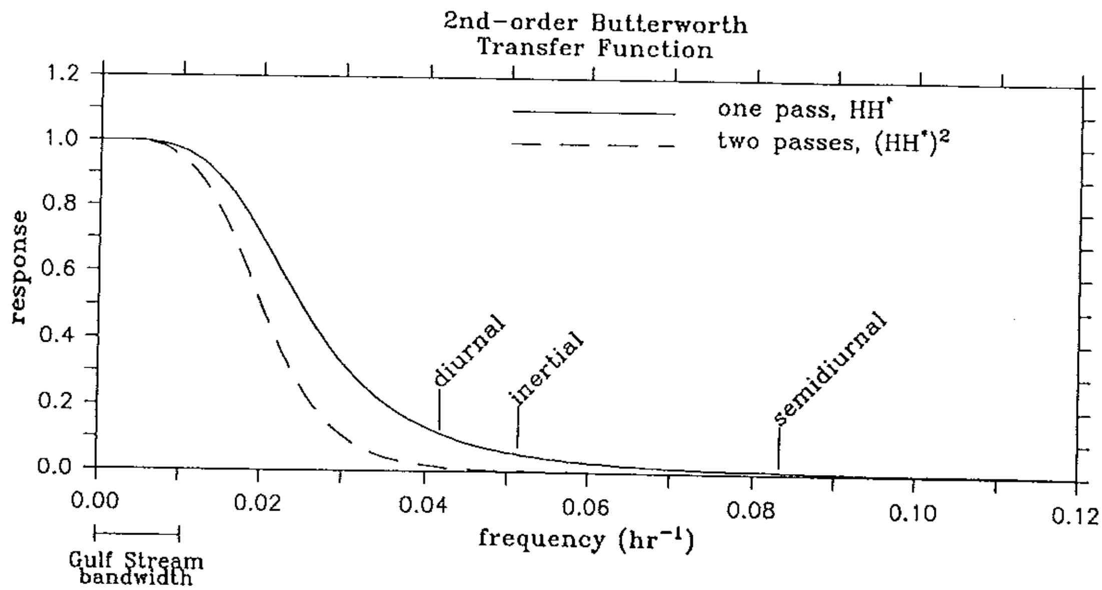



Figure 11: The transfer function for the 2nd-order Butterworth filter is illustrated (solid) along with the effective transfer function (dashed) corresponding to the forward and backward application of the filter. The filter cutoffs are 0.025 $hr^{-1}$ (solid) and 0.02 $hr^{-1}$ (dashed).

FILTER_NAMES calls a 2nd-order recursive filter of the general form given by:

$$y(t_n) = b_0 x(t_n) + b_1 x(t_{n-1}) + b_2 x(t_{n-2}) - a_1 y(t_{n-1}) - a_2 y(t_{n-2}) \tag{8}$$

The recursive filter depends on both the input series to be filtered $(x(t), t \leq t_n))$ and the output $(y(t), (t < t_n))$. The value of $y(t_n)$ depends on $x(t_n)$ and past values of $x(t)$ and $y(t)$; the filter is not symmetric. Since recursive filters are 'one-sided' there is a distortion of the phase relation between the input and the output. This distortion can be removed by filtering twice: once passing forward in time and once passing backward.

The filtering twice affects the overall transfer function of the operation. The order of the combination (forwards and backwards) filtering is double that of a single pass; The transfer function is squared which results in a overall cutoff frequency (half power point) that is reduced relative to the cutoff for the original filter (single pass).

The Butterworth filter design is known for its characteristic sharp monotonic transition between flat pass and stop bands with a minimum of coefficients. The Butterworth is also well known and used regularly in the oceanographic field. FILTER_NAMES is a 2nd-order Butterworth with a cutoff frequency of 40 hr. The equation, for half hour sample spacing, is simply,

$$y(t_n) = 0.0015x(t_n) + 0.0029x(t_{n-1}) + 0.0015x(t_{n-2}) - 1.8890y(t_{n-1}) - 0.8950y(t_{n-2}). \quad (9)$$

As described above, the filter is passed over the data forward and reverse, so the effective order and cutoff frequency are 4 and 0.02004 $hr^{-1}$ (49.89 hr).

Transients at the records' ends are reduced by removing a linear ramp generated from the first and last points of the series before filtering. The same linear ramp is added after filtering. Twenty hours of data at each end of the filtered series are discarded to avoid contamination by startup transients.

The filter's cutoff frequency of 0.02004 $hr^{-1}$ (49.89 hr) suitably removes the tides and inertial motions while preserving the content associated with the Gulf Stream's motion (Figure 11). Over 96% of the variance due to the Gulf Stream is at periods greater than four days, or equivalently $f <$ 0.0104 $hr^{-1}$ (Watts and Johns, 1982).

After filtering, the routine subsamples the time series at six-hour intervals centered on 0000, 0600, 1200, and 1800 UT. A subroutine SUBSAMPLE.M moves through the data in jumps of six hours, however it checks the time of the sample as it procedes and occassionally (usually only once, if at all), it must adjust by one sample (0.5 hr). This adjustment is necessary when the clock drifts such that the sample time of the IES shifts from one side of an hour to another (*e.g.* 02:01 to 01:59).

## CONTROL FILE

The control file for FILTER_NAMES.M is another M-file, NAMES.M. Within NAMES.M arrays are assigned which contain the filter coefficients; the names of the files to be filtered; and, if $\tau$ is being filtered, the calibration parameters (B-intercepts). When FILTER_NAMES calls NAMES the following variables are filled:

**Arrays b and a**

**b** - An array of coefficients which multiply the input time series, as in Equation 8. The coefficients in the equation, $b_0, b_1$, and $b_2$ are b(1)–b(3).

**a** - An array of coefficients which multiply the past output, $y(t)$, as in Equation 8. The coefficients in the equation, $a_0, a_1$, and $a_2$ are a(1)–a(3). Note $a_0 =$ a(1) $= 1$.

**Array z**

**z** - An array of strings containing the names of the files to be filtered. All names must be the same length (Note, in the example below, some are padded with blanks on the left). The string is concatenated with whatever suffix or prefix already incorporated into the 'load' statment within FILTER_NAMES. For example, a suffix may need to be adjusted depending on the input file (*e.g.* '.seacor','.despike_prs', '.despike_tmp').

## Array bints

**bints** - The b-intercepts for calibrating $\tau$ to $Z_{12}$.

MATLAB is case sensitive, thus NAMES.M and FILTER_NAMES.M expect input variables in lower case (see the example NAMES.M M-file listed below). The names within 'z' are strings and must each be enclosed in single quotes as in the example below. Square brackets enclose elements of an array, and semicolons terminate rows. Note that '% ' is a comment character and everything on a line after the percent is considered a comment and is not executed.

The function 'butter' is a Signal Processing Toolbox routine to calculate the filter coefficients. The arguments are the order of the filter and the cutoff frequency (scaled by the nyquist frequency).

```
% Central Array 87-88
% Bints from Z12STAR calibration
%
%                 YR   IES   BINT*          S*
z=[
'PIES88H2';     %88   62   5392.289     16.646
'PIES88H3';     %88   63   2260.729     14.822
' IES88H4';     %88   64   2428.124     33.821
' IES88H5';     %88   65   1171.654     34.532
' IES88I1';     %88   71   1022.707     32.972
'PIES88I2';     %88   72   4719.848     14.738
' IES88I3';     %88   73   6353.378     18.287
' IES88I4';     %88   74   3691.656     21.180
' IES88I5']     %88   75   5548.324      2.139

[b,a,]=butter(2,.025);

bints=[5392.;2261.;2428.;1172.;1023.;4720.;6353.;3692.;5548.]
```

# 3 PROGRAM CODE

## 3.1 BUNS_AUG89.FOR

```
C*****
C****************************************************************************
C*****                                                                      *
C*****  PROGRAM: BUNS_APR88.FOR                                             *
C*****  PURPOSE: To replace the original CARP and BUNS programs.  Reads     *
C*****     the data file created from the cassette reader and interprets   *
C*****     the message codes.  The good data bits are then decoded into     *
C*****     32-bit computer words.                                           *
C*****                                                                      *
c@@@@@  Revised May 1989    BUNS_MAY89.FOR
c@@@@@     Includes the following steps for plotting the data:
c@@@@@     a) writes only every 4th sample to unformatted plot files
c@@@@@     b) 10 lsbs of tau and seqno are written to separate files
c@@@@@        in addition to the full values
c@@@@@
C*****                                                                      *
C*****  I/O UNITS:  KREAD  =  7 --> Input data                              *
C*****              KCTRL  =  8 --> Input control card file                 *
C*****              KLOG   =  9 --> Output log file                         *
C*****              KWRITE = 10 --> Output data (all types)                 *
C*****              KPLOT5 = 11 --> Output PLOT5 data (TT's only)           *
C*****              KSEQ   = 12 --> Output sequence numbers for PLOT5       *
C*****              KPRS   = 13 --> Output pressure data for PLOT5          *
C*****              KTMP   = 14 --> Output temperature data for PLOT5       *
C****************************************************************************
c@@@@@  Revised AUG 1989    BUNS_AUG89.FOR
c***** Modified from buns_may89 so that no "junk" was inserted at the end
c***** of the buns plots.
c*****
      INTEGER*4 OUTREC(100), SAMPLING_PERIOD, mask, lsb(100)
      INTEGER*4 KPLOT5, KSEQ, KPRS, KTMP,klsb,kslsb
      INTEGER*2 INUM, NWDS, NUMBIT
      INTEGER*2 ISHIFT(10), OUTWDS
      INTEGER*4 TESTW(20), TESTV(20)
      INTEGER*4 IW, OW, IT, LASTWD, IPTAU, IPSEQ, IPPRS, IPTMP
      INTEGER*4 DECODE(200), NCASREC
      INTEGER*4 SPANA(5), SPANB(5)
      INTEGER*4 FSTTAU, LSTTAU, SEQPLT, PRSPLT, TMPPLT
      INTEGER*2 EOF
      INTEGER*4 KREAD, KWRITE, KLOG, KCTRL, KFMT
      INTEGER*2 NLCC, NOVFL, NPE, NSR, NLS
      REAL*4 OUTTAU(4800), OUTSEQ(4800), OUTPRS(4800), OUTTMP(4800)
      real*4 taulsb(4800),seqlsb(4800)
      COMMON /CARD1/ KREAD, KWRITE, KLOG, KCTRL, KFMT
      COMMON /CARD1A/ KPLOT5, KSEQ, KPRS, KTMP
      COMMON /CARD2/ NWDS, NSECT, LASTWD, DECODE
      COMMON /CARD2A/ SEQPLT,FSTTAU, LSTTAU, PRSPLT, TMPPLT
      COMMON /CARD3/ ITHROT, NUNSP, SPANA, SPANB, TESTW, TESTV, ISHIFT
      COMMON /CARD6/ NCASREC, NLCC, NOVFL, NPE, NSR, NLS
      DATA KREAD/7/, KCTRL/8/, KLOG/9/, KWRITE/10/, KFMT/0/, KPLOT5/11/
```

```
      DATA KSEQ/12/, KPRS/13/, KTMP/14/
      DATA NCASREC/0/, EOF/0/, IOUTREC/0/
      DATA IPTAU/0/, IPSEQ/0/, IPPRS/0/, IPTMP/0/
      data mask/255/, klsb/15/,kslsb/16/,mask10/1023/
      data itotal/0/, istot/0/
      data  nskip/4/
C*****
C*****  Open the I/O units and files
C*****  Wait until later to open the output dataset according to the
C*****  desired output form of the file (formatted or binary).
C*****  Also open the plot units, if needed, in the S/R CONTROL_CARDS.
C*****
      OPEN(UNIT=kread,STATUS='OLD',BLOCKSIZE=512,
     @     FORM='UNFORMATTED',CARRIAGECONTROL='NONE',
     @     ACCESS='SEQUENTIAL',RECORDTYPE='VARIABLE')
      OPEN(UNIT=kctrl,STATUS='OLD')
      OPEN(UNIT=klog,STATUS='NEW',FORM='FORMATTED')
C*****
C***** Read the control card file
C*****
      CALL CONTROL_CARDS
C*****
      WRITE(*,42)
 42   FORMAT(1X,//,'  PROGRAM IS RUNNING. PLEASE WAIT. RUNNING TIME IS
     @8-12 MINUTES',//)
C*****
C*****                        Main Loop
C***** Process the data for one sampling period at a time and then write it
C***** to the output data set.  You may need to use more than one cassette
C***** record (NSECT) for each sampling period.
C***** Decode GOODBITS into NWDS output words in the array OUTREC.
C*****
   10 CONTINUE
      OW = 0
      IW = 0
      IT = 1
      DO 50 SAMPLING_PERIOD = 1, NSECT
C*****
C*****  Process the next cassette record for this sampling period.
C*****
      CALL NEWCARP(EOF)
      IF (EOF .NE. 0) GO TO 90
      NCASREC = NCASREC + 1
C*****
C***** Decode the words in this section and save them in the output
C***** array. DECODE stores the word lengths of the data.
C***** A negative value in DECODE denotes the end of a
C***** cassette record (which usually means a sample period).
C*****
   15 CONTINUE
      IW = IW + 1
      NUMBIT = DECODE(IW)
      IF (NUMBIT .GE. 0)  THEN
         OW = OW + 1
```

```
            OUTREC(OW) = NXTBIT(NUMBIT)
            IF (IW .LE. LASTWD)  GO TO 15
            WRITE(KLOG,20) LASTWD, IW
   20       FORMAT(/5X,'PROGRAM ERROR - IW EXCEEDS LASTWD AT STMT# 15',
     @      /5X,'LASTWD =',I4,'  IW =',I4,5X,'RUN TERMINATED')
            STOP 15
         END IF
C*******
C******* Check the value of all the test words in this section.
C******* If any test fails - ignore all output words decoded
C******* thus far and start decoding for section 1 again.
C*******
   25 CONTINUE
         IF (TESTW(IT) .GT. 0)  THEN
            IF (TESTW(IT) .LE. OW)  THEN
               I = TESTW(IT)
               IF (OUTREC(I) .NE. TESTV(IT))  THEN
                  WRITE(KLOG,30) TESTW(IT), TESTV(IT), OUTREC(I), NCASREC
   30             FORMAT(/5X,'BAD VALUE FOR OUTPUT WORD#',I4,
     @            ', TESTVALUE =',I6,', ACTUAL VALUE =',I6,
     @            ' - CASSETTE REC#',I4)
                  GO TO 10
               END IF
                 IT = IT + 1
               GO TO 25
            END IF
         END IF
C*****
C***** Testing is finished.
C***** End the decoding loop - repeat NSECT times.
C*****
   50 CONTINUE
C*****
C***** Unspan the words which spanned sections and throw out any
C***** word at position ITHROT.
C*****
         IU = 1
         I = 1
         J = 1
         IF (ITHROT .NE. 0 .OR. NUNSP .NE. 0) THEN
   60        OUTREC(I) = OUTREC(J)
            IF (J .NE. ITHROT) THEN
               IF(J .EQ. SPANA(IU)) THEN
   65             JJ = SPANB(IU)
                  OUTREC(I) = OUTREC(JJ) + (OUTREC(J)*2**ISHIFT(IU))
                  IU = IU + 1
                  J = JJ
               END IF
   70          I = I + 1
            END IF
   75       J = J + 1
            IF (J .LE. LASTWD) GO TO 60
         END IF
C*****
```

```
C***** This should finish one sampling period.
C***** Calculate the number of words to be written out to the disk
C***** and write out to the disk file using the desired format.
C*****
   80 CONTINUE
      OUTWDS = NWDS-NUNSP-NSECT-ITHROT+1
      IOUTREC = IOUTREC + 1
      IF (KFMT .EQ. 0) THEN
         WRITE(KWRITE) (OUTREC(IO), IO=1,OUTWDS)  ! Unformatted output
      ELSE
         WRITE(KWRITE,85) (OUTREC(IO), IO=1,OUTWDS)  ! Formatted output
   85    FORMAT(8I10)
      END IF
C*****
C*****  Store the various data measurements in different output arrays for
C*****  plotting. When the arrays are full, dump them out on disk.
C*****  Only write every 4th sampling period to plotting files
C*****
      nskip = nskip + 1
      if (nskip .lt. 4) go to 10
      nskip = 0
      IF (FSTTAU .NE. 0) THEN
         DO 88 ITAU = FSTTAU, LSTTAU
            IPTAU = IPTAU + 1
            OUTTAU(IPTAU) = OUTREC(itau)
            taulsb(iptau) = jiand(outrec(itau),mask10)
   88    CONTINUE

         IF (IPTAU .EQ. 4800) THEN
             itotal = itotal + iptau
             WRITE(KPLOT5) OUTTAU
             write(klsb) taulsb
             IPTAU = 0
         END IF
      end if
      IF (seqPLT .NE. 0) THEN
         IPSEQ = IPSEQ + 1
         OUTSEQ(IPSEQ) = OUTREC(SEQPLT)
            lsbseq = jiand(outrec(seqplt),mask10)
            seqlsb(ipseq) = lsbseq
         IF (IPSEQ .EQ. 4800) THEN
             istot = istot + ipseq
            WRITE(KSEQ) OUTSEQ
            write(kslsb) seqlsb
            IPSEQ = 0
         END IF
      end if
      IF (PRSPLT .NE. 0) THEN
         IPPRS = IPPRS + 1
         OUTPRS(IPPRS) = OUTREC(26)
         IF (IPPRS .EQ. 4800) THEN
            WRITE(KPRS) OUTPRS
            IPPRS = 0
         END IF
```

```
      END IF
      IF (TMPPLT .NE. 0) THEN
         IPTMP = IPTMP + 1
         OUTTMP(IPTMP) = OUTREC(27)
         IF (IPTMP .EQ. 4800) THEN
            WRITE(KTMP) OUTTMP
            IPTMP = 0
         END IF
      END IF
C*****
C***** Go get next chunk of data.
C*****
      GO TO 10
C*****
C***** End of file condition, wrap things up.  First dump what's stored
C***** in the OUTTAU array.  Note that there may be "junk" at the end of
C***** this array, if less than 4800 new points were used.
C*****
   90 CONTINUE
c*****--------------------------------
c***** beginning of modification made 13-Aug-89
c***** the output arrays are set to zero beyond the real data to avoid
c***** "junk" (referred to below) at the end of the plots.
c*****
do 500 i=iptau,4800
taulsb(i)=0.0
outtau(i)=0.0
500 continue

c*****
c***** note I claim ipseq,ipprs and iptmp should be equal, and therefore
c***** enclose them in a single DO loop.
c*****
do 501 i=ipseq,4800
seqlsb(i)=0.0
outseq(i)=0.0
outprs(i)=0.0
outtmp(i)=0.0
501 continue

c*****
c***** end of modification made 13-Aug-89
c*****--------------------------------


      IF (FSTTAU .NE. 0) then
         WRITE(KPLOT5) OUTTAU
         itotal = itotal + iptau
         write(klsb) taulsb
      End if
      IF (SEQPLT .NE. 0) then
          WRITE(KSEQ) OUTSEQ
          istot = istot + ipseq
          write(kslsb) seqlsb
```

```
      end if
      IF (PRSPLT .NE. 0) WRITE(KPRS) OUTPRS
      IF (TMPPLT .NE. 0) WRITE(KTMP) OUTTMP
      WRITE(KLOG,95) NCASREC, IOUTREC
   95 FORMAT(' END OF DATA ENCOUNTERED'/
     @       ' NUMBER OF CASSETTE RECORDS PROCESSED = ',I6/
     @       ' NUMBER OF OUTPUT DATA RECORDS WRITTEN = ',I6)
      WRITE(KLOG,100) NLCC, NOVFL, NPE, NSR, NLS
  100 FORMAT(/' NUMBER OF DOUBLE LAST CHARACTERS = ',I6/
     @        ' NUMBER OF OVERRUN FLAGS = ',I6/
     @        ' NUMBER OF PARITY ERRORS = ',I6/
     @        ' NUMBER OF SHORT RECORDS = ',I6/
     @        ' NUMBER OF LOW SIGNALS = ',I6)
      WRITE(klog,43) itotal,istot
 43   FORMAT(1X,//,'     **** PROGRAM IS FINISHED!! ***',//
     @        '  itau = ', i10,'     iseq = iprs = itmp = ',i10)
      STOP
      END
C*****
C***********************************************************************
C*****                                                                 *
C*****                           SUBROUTINES                           *
C*****                                                                 *
C***********************************************************************
C*****
C*****
C***********************************************************************
C*****     SUBROUTINE NXTBIT                                           *
C*****                                                                 *
C***** Purpose :                                                       *
C***** To translate a string of 'N' bit integer words                  *
C***** one word at a time, into standard 32 bit integer word           *
C*****                                                                 *
C*****    Input :                                                      *
C*****    A string of bits representing a string of integer            *
C*****    words of varying lengths                                     *
C*****                                                                 *
C*****    Output :                                                     *
C*****    A 32-bit integer word containing one 'N' bit word            *
C*****    from the input string , padded to the left with binary       *
C*****    zeroes if necessary.                                         *
C*****                                                                 *
C*****                                                                 *
C*****    Usage :                                                      *
C*****         A call to this entry point is of the form               *
C*****         IWORD = NXTBIT (NUMBITS) - where IWORD is               *
C*****         a 32-bit integer word which is to contain the next      *
C*****         NUMBITS bits from the bit string being processed.       *
C*****                                                                 *
C*****                                                                 *
C*****    Errors :                                                     *
C*****    If NUMBITS is less than or equal to zero or                  *
C*****    If NUMBITS is greater than 31                                *
C*****    then IWORD is set to -1 (all binary ones)                    *
```

```
C*****    If the total of NUMBITS in all calls to NXTBIT             *
C*****    exceeds IBITS in last call to INEXT, an error              *
C*****    message will be printed.                                   *
C*****                                                               *
C*****                                                               *
C*********************************************************************
C***** BEWARE   NXTBIT is declared as a function subroutine      *****
C*********************************************************************
C*****
      FUNCTION NXTBIT(NBITS)
      IMPLICIT INTEGER*2 (A-Z)
      INTEGER*2 MASKUSED(8)
      BYTE IARRAY(256)
      INTEGER*4 NXTBIT, ANS, NEXTPART
      INTEGER*4 KREAD, KWRITE, KLOG, KCTRL, KFMT
      COMMON /CARD1/ KREAD, KWRITE, KLOG, KCTRL, KFMT
      COMMON /CARD4/ WINX,BINX,IARRAY
      SAVE MASKNEG, MASKUSED
      DATA MASKNEG/'00FF'X/
      DATA MASKUSED/'FF7F'X, 'FF3F'X, 'FF1F'X, 'FF0F'X,
     @              'FF07'X, 'FF03'X, 'FF01'X, 'FF00'X/
C*****
C***** Check for errors in number of bits to process.
C*****
      IF (NBITS.LE.0) THEN              ! ERROR, RETURN -1
         WRITE(KLOG,66)
   66    FORMAT(' NBITS LESS THAN OR EQUAL TO 0 -- NXTBIT SET TO -1')
         NXTBIT = -1
         RETURN
      ELSE IF(NBITS .GT. 31)THEN        ! ERROR, RETURN -1
         WRITE(KLOG,71)
   71    FORMAT('NBITS GREATER THAN 31 -- NXTBIT SET TO -1')
         NXTBIT = -1
         RETURN
      END IF
C*****
C***** Initialize ANS to 0 and PART to the left-most bits of IARRAY(WINX)
C***** IARRAY(WINX) - Current 8-bit string to process.
C***** BINX - The number of bits of IARRAY(WINX) which have already
C*****        been used (don't want to use them again).
C***** BITWNT - Total number of bits needed to create the 32-bit word
C*****
      ANS = 0                             ! New word to decode
      PART = IARRAY(WINX)
C*****
C***** First mask off the 8 MSB's that make PART negative.
C***** Then mask off any bits which have already been used.
C*****
      PART = IIAND(PART, MASKNEG)
      IF (BINX .GT. 0) THEN
         PART = IIAND(PART,MASKUSED(BINX))
      END IF
      PART = IISHFT(PART,BINX)
      BITWNT = NBITS                      ! Total bits needed
```

```
C*****
C***** See if there are enough bits in this word to get the
C***** full 32-bit word.
C*****
   50 CONTINUE
      IF (BITWNT .GT. 8-BINX) THEN
C*****
C***** Not enough bits - then use all of this word and then come back
C***** here to get more bits from the next word.
C*****
         BITNOW = 8 - BINX
         BITWNT = BITWNT - BITNOW
         WINX = WINX + 1
         BINX = 0
C*****
C***** More than enough bits - use only the bits needed.
C*****
      ELSE
         BITNOW = BITWNT
         BITWNT = 0
         BINX = BINX + BITNOW
         IF (BINX .EQ. 8) THEN
            BINX = 0
            WINX = WINX + 1
         END IF
      END IF
C*****
C***** Now have some or all of the bits needed. Right justify them.
C*****
      IF (BITNOW .LT. 8) THEN
          NOWSHFT = 8 - BITNOW
          PART = IISHFT(PART, -NOWSHFT)
      END IF
      NEXTPART = PART
C*****
C***** Shift bits already in ANS to the left to make room for new bits.
C***** Then 'OR' in the new bits.
C*****
      ANS = JISHFT(ANS,(BITNOW))
      ANS = IOR(ANS,NEXTPART)
C*****
C***** Are more bits needed?
C***** If yes: then get the next iarray word. If no: return ANS.
C*****
      IF (BITWNT.GT.0) THEN
         PART = 0
         NEXTPART = 0
         PART = IARRAY(WINX)
         PART = IIAND(PART, MASKNEG)
         GO TO 50
      END IF
      NXTBIT = ANS
      RETURN
      END
```

```
C*****
C***************************************************************************
C*****                                                                     *
C*****  SUBROUTINE:  NEWCARP                                               *
C*****  PURPOSE:  To read the Sea Data Reader data file. Then interpret and *
C*****     remove the message code bits; keep only the data bits.  Process *
C*****     one cassette record at a time.                                  *
C*****                                                                     *
C***************************************************************************
C*****
      SUBROUTINE NEWCARP(EOF)
      IMPLICIT INTEGER*2 (A-Z)
      BYTE DATABITS(256), CODEBITS(256), GOODBITS(200)
      INTEGER*4 KREAD, KWRITE, KLOG, KCTRL, KFMT, NCASREC
      COMMON /CARD1/ KREAD, KWRITE, KLOG, KCTRL, KFMT
      COMMON /CARD4/ WINX, BINX, GOODBITS
      COMMON /CARD6/ NCASREC,NLCC, NOVFL, NPE, NSR, NLS
      SAVE LAST_USED, LSTWD, IBLOCK, MASKNEG
      DATA IBLOCK/0/, LSTWD/256/, LAST_USED/256/
      DATA MASKTYPE/'0080'X/, TYPEA/'0000'X/, MASKLCC/'0001'X/
      DATA MASKPE/'0080'X/, MASKTRACK/'000F'X/, MASKSHORT/'0040'X/
      DATA MASKLOWSIG/'0020'X/, MASKOVFL/'0040'X/
      DATA MASKNEG/'00FF'X/
      DATA NLCC/0/, NOVFL/0/, NPE/0/, NSR/0/, NLS/0/
C*****
C***** Main Processing Loop
C***** The data file is a binary unformatted file of 512-byte blocks
C***** Read one block at a time.
C*****
   10 CONTINUE
      EOR = 0
      IKEEP = 0
      DO 12 L=1,200
         GOODBITS(L) = 0
   12 CONTINUE
      IF (LAST_USED .EQ. 256) THEN
         READ(KREAD, END=60, ERR=15)
     c        (DATABITS(IN), CODEBITS(IN), IN=1,256)
   15    IBLOCK = IBLOCK + 1
         LAST_USED = 0
      END IF
      FRSTWD = LAST_USED + 1
      DO 50 NWORD = FRSTWD, LSTWD
         NOWHIBITS = CODEBITS(NWORD)
         NOWHIBITS = IIAND(NOWHIBITS, MASKNEG)
C*****
C*****  Bits numbered 0 to 15 with LSB = 0 and MSB = 15.
C*****  Determine if a data word (type A - low) or a
C*****  message word (type B - hi) using bit 15.
C*****
         TEST = IIAND(MASKTYPE, NOWHIBITS)
   20    CONTINUE
         IF (TEST .EQ. TYPEA) THEN      ! data word - keep the necessary bits
C*****
```

```
C*****  Check to see if its the last data word of the cassette record.
C*****  Zero out all bits except bit 8; If hi, then last character.
C*****
              TEST = IIAND(MASKLCC,NOWHIBITS)
              IF (TEST .EQ. MASKLCC) THEN
                 IF (IKEEP .EQ. 0) THEN ! Make sure there aren't two in a row
                    WRITE(KLOG,24) NWORD, IBLOCK, NCASREC
   24               FORMAT(' DOUBLE LCC at word = ',I6,
     @                     ' of block = ',I6,' (NCASREC = ',I6,')')
                    NLCC = NLCC + 1
                    GO TO 50
                 END IF
                 EOR = 1   ! end of record encountered
              END IF
C*****
C*****  Check to see if there is an overrun flag, indicating at least one
C*****  missed scan of data. Zero out all bits except bit 14.
C*****  If hi, then overrun has occurred.
C*****
              TEST = IIAND(MASKOVFL,NOWHIBITS)
              IF (TEST .EQ. MASKOVFL) THEN
                 WRITE(KLOG,25) NWORD, IBLOCK, NCASREC
   25            FORMAT(' OVERRUN FLAG at word = ',I6,
     @                  ' of block = ',I6,' (NCASREC = ',I6,')')
                 NOVFL = NOVFL + 1
              END IF
C*****
C*****  Keep the data dits , get rid of the code bits
C*****
              IKEEP = IKEEP + 1
              GOODBITS(IKEEP) = DATABITS(NWORD)
C*****
C*****  Processing Type B - message words
C*****
           ELSE
              NOWLOWBITS = DATABITS(NWORD)
              NOWLOWBITS = IIAND(NOWLOWBITS, MASKNEG)
C*****
C*****  Test for pariety errors.  If bit 7 is hi, errors occurred.
C*****
   30         TEST = IIAND(MASKPE, NOWLOWBITS)
              IF (TEST .EQ. MASKPE) THEN  ! Parity error occurred
                 TEST = IIAND(MASKTRACK, NOWLOWBITS)
                 WRITE(KLOG,35) TEST, NWORD, IBLOCK,NCASREC
   35            FORMAT(' PARITY ERROR = ',I6,' at word = ',I6,
     @                  ' of block = ',I6,' (NCASREC = ',I6,')')
                 NPE = NPE + 1
              END IF
C*****
C*****  Test for a short record. If bit 6 is hi, then yes.
C*****
              TEST = IIAND(MASKSHORT, NOWLOWBITS)
              IF (TEST .EQ. MASKSHORT) THEN   ! Record was short
                 WRITE(KLOG,40) NWORD, IBLOCK, NCASREC
```

```
 40         FORMAT(' SHORT RECORD at word = ',I6,
     @              ' of block = ',I6,' (NCASREC = ',I6,')')
            NSR = NSR + 1
            IF (IKEEP .NE. 0) EOR = 1
         END IF
C*****
C*****  Test for low signal strengthduring record. If bit 5 is hi, then yes.
C*****
         TEST = IIAND(MASKLOWSIG, NOWLOWBITS)
         IF (TEST .EQ. MASKLOWSIG) THEN   ! Signal strength was weak
            WRITE(KLOG,45) NWORD, IBLOCK, NCASREC
 45         FORMAT(' WEAK SIGNAL encountered at word = ', I6,
     @              ' of block = ',I6,' (NCASREC = ',I6,')')
            NLS = NLS + 1
         END IF
C*****
C*****  Check to see if there is an overrun flag, indicating at least one
C*****  missed scan of data. Zero out all bits except bit 14.
C*****  If hi, then overrun has occurred.
C*****
         TEST = IIAND(MASKOVFL,NOWHIBITS)
         IF (TEST .EQ. MASKOVFL) THEN
            WRITE(KLOG,25) NWORD, IBLOCK, NCASREC
            NOVFL = NOVFL + 1
         END IF
      END IF
C*****
C*****  Finished interpreting this word. If EOR (end of cassette record),
C*****  return to main program to decode into 32-bit computer words.
C*****
      IF (EOR .NE. 0) THEN
          LAST_USED = NWORD
          WINX = 1
          BINX = 0
          RETURN
      END IF
 50 CONTINUE
C*****
C*****  Finished with this block of data, get the next one.
C*****
    LAST_USED = 256
    GO TO 12
C*****
C***** End of data encountered.
C*****
 60 CONTINUE
    EOF = -1
    WRITE(KLOG,65) IBLOCK
 65 FORMAT(//5X,'END OF FILE ENCOUNTERED FOLLOWING IBLOCK # ',I5//)
    RETURN
    END
C*****
C***********************************************************************
C*****
```

```
      SUBROUTINE CONTROL_CARDS
      INTEGER*2 IDI, IDS(5), NWDS
      INTEGER*4 IVALS(10), NSECT
      INTEGER*4 NEG1CT, DECODE(200), IW
      INTEGER*4 TESTW(20),TESTV(20), IT
      INTEGER*4 ITHROT, SPANA(5), SPANB(5), IU
      INTEGER*4 LASTWD, NUNSP
      INTEGER*4 FSTTAU, LSTTAU, SEQPLT, PRSPLT, TMPPLT
      INTEGER*2 ISHIFT(10)
      INTEGER*4 KREAD, KWRITE, KLOG, KCTRL, KFMT
      INTEGER*4 KPLOT5, KSEQ, KPRS, KTMP
      COMMON /CARD1/ KREAD, KWRITE, KLOG, KCTRL, KFMT
      COMMON /CARD1A/ KPLOT5, KSEQ, KPRS, KTMP
      COMMON /CARD2/ NWDS, NSECT, LASTWD, DECODE
      COMMON /CARD2A/ SEQPLT, FSTTAU, LSTTAU, PRSPLT, TMPPLT
      COMMON /CARD3/ ITHROT, NUNSP, SPANA, SPANB, TESTW, TESTV, ISHIFT
      DATA IDS/'NW', 'WL', 'SV', 'US', 'WF'/
      DATA IW/0/, NEG1CT/0/, LASTWD/0/
      DATA TESTW/20*0/, TESTV/20*0/, IT/0/
      DATA ITHROT/0/, SPANA/5*0/, SPANB/5*0/, IU/0/, NUNSP/0/
C*****
C***** Read in the control parameters and set error options.
C*****
      WRITE(KLOG,10)
   10 FORMAT(5X,' CONTROL CARDS FOR DECODING'/)
C*****
C*****     Reading loop for the the control cards
C*****
   20 CONTINUE
      READ(KCTRL,22,END=65) IDI, (IVALS(I),I=1,10)
   22 FORMAT(A2,3X,10I5)
      WRITE(KLOG,24) IDI, (IVALS(I),I=1,10)
   24 FORMAT(/5X,A2,2X,10I10)
C*****
C***** Check for a 'NUMBER OF WORDS' (NW) card.
C***** NSECT - The number of cassette records used to hold
C***** all the data from one sampling period.  Usually this
C***** is one.
C***** NWDS - Number of words to be decoded, this does not
C***** include the -1 word.
C*****
      IF (IDI .EQ. IDS(1))  THEN
         NWDS = IVALS(1)
         NSECT = IVALS(2)
         IF (NSECT .EQ. 0)  NSECT = 1
C*****
C***** Check for a word length (WL) card.
C***** Save the word lengths in array 'DECODE' - Ignore zero values
C***** -1 on this card flags the end of the cassette record.
C*****
      ELSE IF (IDI .EQ. IDS(2))  THEN
         DO 30 I = 1, 10
           IF (IVALS(I) .NE.  0)  THEN
              IF (IVALS(I) .LT.  0)  NEG1CT = NEG1CT + 1
```

```
                IW = IW + 1
                DECODE(IW) = IVALS(I)
             END IF
   30     CONTINUE
C*****
C***** Check for a 'SPECIAL VALUE' (SV) control card.
C***** Save the testwords in TESTW and the testvalues in TESTV.
C***** Ignore negative and zero values for testwords.
C*****
        ELSE IF (IDI .EQ. IDS(3))  THEN
           DO 40 I = 1, 10, 2
             IF (IVALS(I) .GT. 0)  THEN
                IT = IT + 1
                TESTW(IT) = IVALS(I)
                TESTV(IT) = IVALS(I+1)
             END IF
   40     CONTINUE
C*****
C***** Check for an 'UNSPAN' (US) words control card.
C***** First value is word to be thrown out (0=none).
C***** Other pairs of values are words to be unspanned.
C***** Save the values in SPANA and SPANB. Ignore zero values.
C*****
        ELSE IF (IDI .EQ. IDS(4)) THEN
           ITHROT = IVALS(1)
           DO 50 I=2,9,2
             IF(IVALS(I) .GT. 0) THEN
                IU = IU + 1
                SPANA(IU) = IVALS(I)
                SPANB(IU) = IVALS(I+1)
             END IF
 50       CONTINUE
C*****
C***** Check for a 'WRITE FORMAT' (WF) control card for
C***** output data on unit KWRITE.
C***** IF 0 -> UNFORMATTED     IF 1 -> FORMATTED
C***** Open the output file accordingly.
C***** Save the word numbers of the first and last travel times for plotting.
C***** Open all plotting file units if needed.
C*****
        ELSE IF (IDI .EQ. IDS(5)) THEN
           KFMT = IVALS(1)
           IF (KFMT .EQ. 0) THEN
              OPEN(UNIT=kwrite,STATUS='NEW',FORM='UNFORMATTED')
           ELSE
              OPEN(UNIT=kwrite,STATUS='NEW',FORM='FORMATTED')
           END IF
           SEQPLT = IVALS(2)
           IF (SEQPLT .NE. 0) then
              OPEN(UNIT=kseq, STATUS='NEW', FORM='UNFORMATTED')
              OPEN(UNIT=kslsb,STATUS='NEW',FORM='UNFORMATTED')
           end if
           FSTTAU = IVALS(3)
           LSTTAU = IVALS(4)
```

```
          IF (FSTTAU .NE. 0) then
             OPEN(UNIT=kplot5, STATUS='NEW', FORM='UNFORMATTED')
             OPEN(UNIT=klsb,STATUS='NEW',FORM='UNFORMATTED')
          end if
          PRSPLT = IVALS(5)
          IF (PRSPLT .NE. 0)
     @       OPEN(UNIT=kprs, STATUS='NEW', FORM='UNFORMATTED')
          TMPPLT = IVALS(6)
          IF (TMPPLT .NE. 0)
     @       OPEN(UNIT=ktmp, STATUS='NEW', FORM='UNFORMATTED')
C*****
C***** If control card is none of the above then it is invalid.
C***** So print a message and terminate the run.
C*****
       ELSE
          WRITE(KLOG,55)
   55     FORMAT(/5X,'PRECEEDING CONTROL CARD HAS AN INVALID I.D.',
     @    ' - RUN TERMINATED')
          STOP 55
       END IF
       GO TO 20
C*****
C***** Last control card has been read; Set parameters.
C*****
   65 CONTINUE
       LASTWD = IW
       NUNSP = IU
C*****
C***** If there are words to be unspanned, get the multiplicative
C***** values from the associated word lengths.  Assume normal bit
C***** order, LSB's to the right (higher word #).
C***** Note that the DECODE array still contains the ITHROT words
C***** so that IWD is calculated to skip over these values.
C*****
       IF(NUNSP .NE. 0) THEN
          DO 70 I=1,NUNSP
          IWD = SPANB(I) + I
          ISHIFT(I) = DECODE(IWD)
  70      CONTINUE
       END IF
C*****
C***** Check for control card consistency.
C*****
  75  CONTINUE
       IF (NSECT .NE. NEG1CT)  THEN
          WRITE(KLOG,78) NSECT, NEG1CT
  78      FORMAT(/5X,'NW CARD SPECIFIES',I5,
     @    ' SECTIONS BUT WL CARDS CONTAIN',I5,
     @    ' SECTION END MARKERS (NEG) - RUN TERMINATED')
          STOP 78
       END IF
   80 CONTINUE
       I = IW - NEG1CT
       IF (NWDS .NE. I)  THEN
```

```
          WRITE(KLOG,82) NWDS, I
   82      FORMAT(/5X,'NW CARD SPECIFIES',I5,
     @    ' WORDS BUT WL CARDS CONTAIN',I5,
     @    ' (NON-NEGATIVE NON-ZERO) WORDS - RUN TERMINATED')
          STOP 82
       END IF
C*****
C***** Control cards were okay. Return to main program.
C*****
   85 CONTINUE
       RETURN
       END
```

## 3.2 PUNS_MAY88.FOR

```
C*******
C*******                          PUNS_MAY88.FOR
C*******
C*******     THIS PROGRAM WILL PRODUCE HISTOGRAMS AND/OR LISTINGS
C******* OF TRAVEL TIME BURSTS OF A SPECIFIED RANGE OF SAMPLES.
C******* THE LISTINGS ARE EITHER IN INTEGER COUNTS OR CONVERTED TO THE
C******* DECIMAL EQUIVALENTS.  THREE TYPES OF HISTOGRAMS CAN BE
C******* PRODUCED:  1) LEVEL-1 (L1) ARE INDIVIDUAL PLOTS OF EACH
C******* SAMPLING PERIOD.  2) LEVEL-2 (L2) ARE PLOTS OF GROUPS OF
C******* RECORDS, SAMPLED AT A GIVEN INTERVAL.  3) LEVEL-3 PRODUCES A
C******* HISTOGRAM OF ALL RECORDS WITHIN THE RANGE SPECIFIED.  A LEVEL-3
C******* PLOT IS PRODUCED EACH AUTOMATICALLY WHEN THE PROGRAM IS
C******* EXECUTED.
C*******      THE USER SPECIFIES THE TOTAL RANGE OF RECORDS TO PROCESS.
C******* IF LEVEL-2 PLOTS ARE TO BE MADE, THE USER MUST ALSO SPECIFY THE
C******* NUMBER OF RECORDS TO USE FOR A HISTOGRAM (GRPSIZ) AND THE
C******* NUMBER OF RECORDS TO SKIP BETWEEN CONSECUTIVE PLOTS (RATE).
C******* THE UPPER AND LOWER BOUNDS OF HISTOGRAMS ARE ALSO SPECIFIED BY
C******* THE USER, THUS ENLARGEMENTS OF A NARROWER RANGE CAN BE MADE.
C******* IF THERE ARE ADDITIONAL SENSORS, SUCH AS PRESSURE, THEIR VALUES
C******* IN COUNTS ARE PRINTED WHEN SAMPLES ARE LISTED.  IF THERE ARE 2
C******* TT DETECTORS, BOTH WILL BE PLOTTED.
C*******
C******* FORTRAN UNIT NUMBERS DESIGNATED AS FOLLOWS:
C*******     KR     (UNIT 5) CONTROL CARD INPUT FILE
C*******     KW     (UNIT 6) PRINTER OUTPUT LOG FILE OF HISTOGRAMS
C*******     KWDEC  (UNIT 7) FLOATING POINT OUTPUT OF SCALED DATA
C*******     KWINT  (UNIT 8) INTEGER OUTPUT OF SCALED DATA
C*******     KRBUNS (UNIT 9) INTEGER INPUT OF BUNS.REV82 DATA
C*******
C*******
      NAMELIST/CARD1/ HEADR
      NAMELIST/CARD2/ NTT, TTYPE
      NAMELIST/CARD3/ NWORDS, LBURST, LBFST, RDFMT
      NAMELIST/CARD4/ NSEN, SENSOR, SWDNO
      NAMELIST/CARD5/ SF1, SF2
      NAMELIST/CARD6/ LBNDA, UBNDA, LBNDB, UBNDB
      NAMELIST/CARD7/ START, END, RATE, GRPSIZ, SEQINC
      NAMELIST/CARD8/ OPTN
      COMMON FREQ,LCT,UCT,LBND,UBND,NUMLN,NUMPL,SF,RATE,GRPSIZ
      COMMON/NUMBR/NTT,LBFST,LBLST,TTYPE
      COMMON/UNIT/KR, KW, KWDEC, KWINT, KRBUNS, RDFMT
      CHARACTER*60 HEADR
      CHARACTER*3 TTYPE(2)
      CHARACTER*2 SENSOR(3), OPTN(4), PR, TP, AM
      CHARACTER*2 DE, INT, L1, L2
      INTEGER*4 FREQ(55,6), FREQA(55), FREQB(55)
      INTEGER*4 LCT(6), UCT(6), LCTA, LCTB, UCTA, UCTB
      INTEGER*4  NTT
      INTEGER*4 NWORDS, LBURST, LBFST, RDFMT
      INTEGER*4 NSEN, SWDNO(3)
      INTEGER*4 LBND(2), UBND(2), LBNDA, LBNDB, UBNDA, UBNDB
```

```
      INTEGER*4 START, END, RATE, RATCTR, GRPSIZ, GRPEND, SEQINC
      INTEGER*4 DESW, INSW, GRL1, GRL2
      INTEGER*4 PRSW,TPSW,AMSW,PRWDNO,TPWDNO,AMWDNO
      INTEGER*4 LEVEL1, LEVEL2, LEVEL3
      INTEGER*4 TO, FROM
      INTEGER*4 NUMLN(2), NUMPL(2), NUMPLA, NUMPLB
      INTEGER*4 IN(100), OUT(100), RECIN, RECOUT, SEQNO
      INTEGER*4 PRESS, TEMP, AMBNS
      INTEGER*4 BOTA,TOPA,BOTB,TOPB
      REAL*4    SF1, SF2, SF(2)
      REAL*4    DOUT(100), DLBNDA, DUBNDA, DLBNDB, DUBNDB
      EQUIVALENCE (FREQ(1,1),FREQA(1)),(FREQ(1,2),FREQB(1)),
     @            (LCT(1),LCTA),(LCT(2),LCTB),
     @            (UCT(1),UCTA),(UCT(2),UCTB),
     @            (LBND(1),LBNDA),(LBND(2),LBNDB),
     @            (UBND(1),UBNDA),(UBND(2),UBNDB),
     @            (NUMPL(1),NUMPLA),(NUMPL(2),NUMPLB),
     @            (SF(1),SF1),(SF(2),SF2)
      PARAMETER (DE='DE',INT='IN',L1='L1',L2='L2')
      PARAMETER (PR='PR',TP='TP',AM='AM')
      DATA KR/5/, KW/6/, KWDEC/7/, KWINT/8/, KRBUNS/9/
      DATA RECIN/0/,RECOUT/0/,SEQNO/-1/,RATCTR/0/
      DATA DESW/0/,INSW/0/,GRL1/0/,GRL2/0/
      DATA PRESS/-99/,TEMP/-99/,AMBNS/-99/,NTT/1/
      DATA TTYPE/2*'    '/,SENSOR/3*'  '/,SWDNO/0,0,0/
      DATA OPTN /4*' '/
C*******
C******* SOME OF THE VARIABLES:
C******* GRPSIZ - THE NUMBER OF CONSECUTIVE BURSTS TO BE SAMPLED
C******* RATE   - THE NUMBER OF BURSTS SKIPPED BETWEEN GROUPS
C******* SEQUENCE NUMBER INCREMENT:
C*******             1 = 15 MIN SAMPLING
C*******             2 = 30 MIN SAMPLING
C*******             4 = 60 MIN SAMPLING
C*******             8 = 120 MIN SAMPLING
C*******
C******* TYPICALLY SF1=20480.0,SF2=20480.0
C******* LOWER AND UPPER BOUNDS ARE SPECIFIED IN TERMS OF COUNTS,
C*******   HENCE, TO FIND ALLOWED RANGE IN ENGINEERING UNITS, DIVIDE
C*******   COUNTS BY SCALE FACTOR:  E.G. 202752./20480.= 9.9 SECONDS
C******* NTT - NUMBER OF DIFFERENT TRAVEL TIME DETECTORS (TTYPE) USED.
C******* NWORDS - NUMBER OF WORDS PER BUNS OUTPUT RECORD (INCLUDING
C*******   SEQ#, LBURST,PRESS,TEMP,AMB)
C******* RDFMT - FORMAT OF INPUT DATASET:  IF 1 => FORMATTED READ
C*******                                   IF 0 => BINARY READ
C*******
C******* INITIALIZE COMMON VARIABLES
C*******
      DO 16 I = 1, 6
      DO 15 J = 1,55
      FREQ(J,I) = 0
   15 CONTINUE
      UCT(I) = 0
      LCT(I) = 0
```

```
   16 CONTINUE
C*****
C*****
      WRITE(*,42)
   42 FORMAT(1X,//,'  THE PROGRAM IS NOW RUNNING!',//,
     @'  THIS MAY TAKE A FEW MINUTES SO SIT BACK AND RELAX.',/)
C*******
C******* READ AND PRINT THE CONTROL CARD INFORMATION
C*******
      READ (KR,NML=CARD1)
      READ (KR,NML=CARD2)
      READ (KR,NML=CARD3)
      READ (KR,NML=CARD4)
      READ (KR,NML=CARD5)
      READ (KR,NML=CARD6)
      READ (KR,NML=CARD7)
      READ (KR,NML=CARD8)
C*******
C******* OPEN THE INPUT BUNS DATA SET FOR READING DEPENDING ON THE
C******* FORMAT OF THE DATA
C*******
      IF (RDFMT .EQ. 0) THEN
         OPEN(UNIT=KRBUNS, STATUS='OLD', FORM='UNFORMATTED')
      ELSE
         OPEN(UNIT=KRBUNS, STATUS='OLD', FORM='FORMATTED')
      END IF
C*******
C******* RESET VARIABLES, IF NECESSARY, TO MAKE SURE THEY ARE CORRECT
C*******
      IF (END .LT. 1)  END = 2**30
      IF (GRPSIZ .LE. 0)  GRPSIZ = 1
      IF(SEQINC .LE. 0)  SEQINC = 1
C*******
C******* SET OPTION SWITCHES FOR THE DESIRED OUTPUT TYPES
C*******
      DO 17 I = 1, 4
      IF (OPTN(I) .EQ. INT) INSW = 1
      IF (OPTN(I) .EQ. DE) DESW = 1
      IF (OPTN(I) .EQ. L1) GRL1 = 1
      IF (OPTN(I) .EQ. L2) GRL2 = 1
   17 CONTINUE
      IF(NSEN .EQ. 0) GO TO 26
      DO 25 I = 1, 3
      IF(SENSOR(I) .NE. PR) GO TO 23
      PRSN = 1
      PRWDNO = SWDNO(I)
      GO TO 25
   23 IF(SENSOR(I) .NE. TP) GO TO 24
      TPSN = 1
      TPWDNO = SWDNO(I)
      GO TO 25
   24 IF(SENSOR(I) .NE. AM) GO TO 25
      AMSN = 1
      AMWDNO = SWDNO(I)
```

```
   25 CONTINUE
C*******
C******* DETERMINE THE REMAINING CONTROLLING VARIABLES
C******* THESE ARE BASED ON THE TYPE OF HISTOGRAMS WANTED
C*******
   26 GRPEND = RATE + GRPSIZ
      LEVEL3 = (GRL1 + GRL2) * 2 + 1
      LEVEL2 = (GRL1 + GRL2) * 2 - 1
      LEVEL1 = GRL1
      LBLST = LBFST + LBURST - 1
      IF(NTT .GT. 1) LBLST = LBFST + (2*LBURST) - 1
C*******
C*******   WRITE HEADER INFO TO LOG
C*******
      WRITE(KW,301) HEADR
  301 FORMAT('1',A60)
      WRITE(KW,303) (TTYPE(I),I=1,NTT),(SENSOR(II),II=1,3)
  303 FORMAT('0',' TYPES OF SENSORS USED: ',5(A4,2X))
      WRITE(KW,305) TTYPE(1)
  305 FORMAT('0',A4,' DETECTOR: ')
      IF(NTT .GT. 1) WRITE(KW,307) TTYPE(2)
  307 FORMAT('+',T60,A4,' DETECTOR: ')
      WRITE(KW,309) SF1
  309 FORMAT(6X,'SCALE FACTOR A = ',F16.5)
      IF(NTT .GT. 1) WRITE(KW,311) SF2
  311 FORMAT('+',T60,5X,'SCALE FACTOR B = ',F16.5)
      WRITE(KW,313) LBNDA,UBNDA
  313 FORMAT(6X,'LBNDA = ',I8,3X,'UBNDA = ',I8)
      IF(NTT .GT. 1) WRITE(KW,315) LBNDB,UBNDB
  315 FORMAT('+',T60,5X,'LBNDB = ',I8,3X,'UBNDA = ',I8)
      WRITE (KW,317) START,END,RATE,GRPSIZ,SEQINC
 317  FORMAT(/10X,'REC #',I6,' THRU ',I6,' WILL BE PROCESSED',
     @       //10X,'SAMPLE RATE =',I6,5X,'GROUP SIZE =',I6,
     @       //10X,'SEQUENCE NO. INC. = ',I6)
      WRITE (KW,319) (OPTN(I),I=1,4)
 319  FORMAT(/10X,'OPTIONS = ',4(A2,2X))
C*******
C*******   FIGURE THE RANGE OF EACH GRAPH LINE & THE # OF LINES / GRAPH
C*******
      DO 30 I = 1,NTT
      UBRNGE = UBND(I) - LBND(I)
      NUMPL(I) = UBRNGE / 50 + 1
      NUMLN(I) = UBRNGE / NUMPL(I) + 1
  30  CONTINUE
C*******
C*******   CONVERT THE UPPER AND LOWER BOUNDS TO DECIMAL SECONDS
C*******
      DLBNDA = LBNDA / SF1
      DUBNDA = UBNDA / SF1
      IF(NTT .EQ. 1) GO TO 35
      DLBNDB = LBNDB / SF2
      DUBNDB = UBNDB / SF2
C*******
C*******   READ THE INPUT DATA FILE, CHECK FOR EOF, INCREMENT COUNTER
```

```
C*******
   35 CONTINUE
      CALL RDBUNS(NWORDS,IN)
      IF (IN(1) .EQ. -1)  GO TO 1000
      RECIN = RECIN + 1
C*******
C******* CHECK WHETHER RECORD SHOULD BE PROCESSED
C******* A)  OUTSIDE RANGE OF RECORDS TO PROCESS
C*******
      IF (RECIN .LT. START)  GO TO 35
      IF (RECIN .GT. END)  GO TO 1100
C*******
C******* B)  DOING LEVEL-1 OR LEVEL-3 PLOTS, USE THIS RECORD
C*******
      IF (RATE .LT. 1)     GO TO 40
C*******
C******* C)  DOING ONLY A LEVEL-2 PLOT, CHECK IF WITHIN GROUP
C*******     TO PROCESS OR TO SKIP.  IF IN GROUP TO PROCESS,
C*******     DO YOU HAVE THEM ALL?  IF SO, THEN RESET COUNTER.
C*******
      RATCTR = RATCTR + 1
   37 IF (RATCTR .LT. RATE) GO TO 35
      IF (RATCTR .LT. GRPEND) GO TO 40
      RATCTR = RATCTR - RATE
C*******
C*******   GENERATE A LEVEL-2 (GROUP) GRAPH IF REQUESTED
C*******
      IF (GRL2.EQ.0) GO TO 37
      TO = RECIN - 1
      FROM = RECIN - GRPSIZ
      CALL FREQGR (LEVEL2,FROM,TO)
      GO TO 37
C*******
C*******   CHECK FOR SEQUENCE ERRORS IN THE FILE
C*******   AND RENAME THE DATA VALUES
C*******   ASSUMES THAT SEQNO IS FIRST WORD AND THE TT'S ARE GROUPED
C*******
   40 CONTINUE
      IF (IN(1) .NE. SEQNO+SEQINC)  WRITE(KW,335) RECIN,IN(1),SEQNO
  335 FORMAT(/10X,'REC #',I6,' => SEQ #',I6,'    RECORD OUT OF',
     @ ' SEQUENCE (FORMER SEQ # WAS',I6,')')
      SEQNO = IN(1)
      IF(PRSN .EQ. 1) PRESS = IN(PRWDNO)
      IF(TPSN .EQ. 1) TEMP = IN(TPWDNO)
      IF(AMSN .EQ. 1) AMBNS = IN(AMWDNO)
      DO 45 L=LBFST,LBLST
      OUT(L)=IN(L)
   45 CONTINUE
C*******
C*******   IF REQUESTED, SCALE TTA AND TTB TO DECIMAL SECONDS AND PRINT
C*******
      IF (DESW .EQ. 0) GO TO 50
      DOUT(1) = SEQNO
      DO 46 I = LBFST,LBLST,NTT
```

```
      DOUT(I) = OUT(I) / SF1
      IF(NTT .GT. 1) DOUT(I+1) = OUT(I+1) / SF2
   46 CONTINUE
      WRITE(KWDEC,410) RECIN,SEQNO,PRESS,TEMP,AMBNS,
     C (DOUT(I),I=LBFST,LBLST)
  410 FORMAT(5I10,/(8F10.5))
C*******
C*******   IF REQUESTED, PRINT THE INTEGER DATA
C*******
   50 CONTINUE
      IF (INSW .EQ. 0)  GO TO 60
      WRITE(KWINT,420) RECIN,SEQNO,PRESS,TEMP,AMBNS,
     C  (OUT(I),I=LBFST,LBLST)
  420 FORMAT(5I10,/(8I10))
C*******
C*******   DETERMINE THE FREQUENCY DISTRIBUTION OF THE DATA
C*******   AND LIMIT THE RANGE
C*******
   60 CONTINUE
      BOTA = 0
      TOPA = 0
      BOTB = 0
      TOPB = 0
      DO 66 I = LBFST,LBLST,NTT
      IF (OUT(I) .GT. LBNDA)  GO TO 62
      OUT(I) = LBNDA
      DOUT(I) = DLBNDA
      BOTA = BOTA + 1
   62 IF (OUT(I) .LT. UBNDA)  GO TO 64
      OUT(I) = UBNDA
      DOUT(I) = DUBNDA
      TOPA = TOPA + 1
   64 CONTINUE
      INDX = (OUT(I) - LBNDA) / NUMPLA + 1
      FREQA(INDX) = FREQA(INDX) + 1
   66 CONTINUE
C*******
C******* IF MORE THAN ONE DETECTOR WAS USED, CALCULATE THE FREQUENCY
C******* DISTRIBUTION OF THE SECOND MEASUREMENTS.  INCREMENT COUNTERS.
C*******
      IF(NTT .EQ. 1) GO TO 76
      LB1 = LBFST + 1
      DO 74 I = LB1,LBLST,NTT
      IF (OUT(I) .GT. LBNDB)  GO TO 70
      OUT(I) = LBNDB
      DOUT(I) = DLBNDB
      BOTB = BOTB + 1
   70 IF (OUT(I) .LT. UBNDB)  GO TO 72
      OUT(I) = UBNDB
      DOUT(I) = DUBNDB
      TOPB = TOPB + 1
   72 CONTINUE
      INDX = (OUT(I) - LBNDB) / NUMPLB + 1
      FREQB(INDX) = FREQB(INDX) + 1
```

```
   74  CONTINUE
       LCTB = LCTB + BOTB
       UCTB = UCTB + TOPB
C*******
C*******   INCREMENT THE COUNTERS
C*******
    76 RECOUT = RECOUT + 1
       LCTA = LCTA + BOTA
       UCTA = UCTA + TOPA
C*******
C*******   GENERATE A LEVEL-1 (SINGLE RECORD) GRAPH IF REQUESTED
C*******
    78 IF (GRL1 .EQ. 0)  GO TO 35
       FROM = RECIN
       TO = 0
       CALL FREQGR(LEVEL1,FROM,TO)
       GO TO 35
C*******
C*******   WRAP UP  -  WRITE OUT MESSAGES TO USER
C*******
  1000 CONTINUE
       WRITE(KW,340)
   340 FORMAT(/10X,'PROCESSING ENDED AT END OF DATA')
  1100 CONTINUE
       WRITE(KW,345) RECIN,RECOUT
   345 FORMAT(///T10,I8,' LOGICAL RECORDS READ',
      @//T10,I8,' LOGICAL RECORDS PROCESSED')
C*******
C*******        TO END:
C*******   PRINT AND GRAPH THE FREQUENCY DISTRIBUTION
C*******   GENERATE A LEVEL-3 (TOTAL) GRAPH
C*******
       CALL FREQGR (LEVEL3,START,RECIN)
       WRITE(*,43)
    43 FORMAT(1X,//,'  WE ARE NOW DONE! - GOOD LUCK!',/)
       STOP
       END
C*******
C************************************************************************
C*******
C*******                          SUBROUTINES
C*******
C************************************************************************
C*******
       SUBROUTINE RDBUNS(NWORDS,IN)
       COMMON/UNIT/KR, KW, KWDEC, KWINT, KRBUNS, RDFMT
       INTEGER*4 IN(100),RDFMT
C*******
C*******   READ EITHER BINARY OR FORMATTED DATA
C*******
       IF(RDFMT .EQ. 1) GO TO 90
       READ(KRBUNS,END=99)(IN(I),I=1,NWORDS)
       RETURN
    90 READ(KRBUNS,95,END=99) (IN(I),I=1,NWORDS)
```

```
   95 FORMAT(8I10)
      RETURN
   99 IN(1)=-1
  100 CONTINUE
      RETURN
      END
C*******
C**********************************************************************
C*******
      SUBROUTINE FREQGR(LEVEL,FROM,TO)
      COMMON FREQ,LCT,UCT,LBND,UBND,NUMLN,NUMPL,SF,RATE,GRPSIZ
      COMMON/NUMBR/NTT,LBFST,LBLST,TTYPE
      COMMON/UNIT/KR, KW, KWDEC, KWINT, KRBUNS, RDFMT
      INTEGER*4 FREQ(55,6),LCT(6),UCT(6)
      INTEGER*4 LBND(2),UBND(2),NUMLN(2),NUMPL(2)
      REAL*4    SF(2)
      INTEGER*4 RATE,GRPSIZ,TTYPE(2)
      INTEGER*4 LEVEL,FROM,TO
      INTEGER*2 LINE(110)
      DATA LINE/110*'X'/
C*******
C*******   GRAPH BOTH THE A AND B TRAVEL TIMES
C*******
      DO 50 K = 1,NTT
      L = LEVEL - 1 + K
C*******
C*******
C*******   PRINT THE GRAPH HEADING
C*******
      WRITE(KW,101) TTYPE(K),FROM
  101 FORMAT(1H1/T50,A4,' FREQUENCY DISTRIBUTION',T100,'REC #',I6)
      IF (TO .NE. 0)  WRITE(KW,102) TO
  102 FORMAT(1H+,T112,'THRU',I6)
      IF (RATE .NE. 0)  WRITE(KW,103) RATE,GRPSIZ
  103 FORMAT(T100,'RATE ',I6,T112,'GROUP',I5)
C*******
C*******   DETERMINE THE MAXIMUM VALUE TO BE GRAPHED
C*******
      MAX = 0
      DO 10 I = 1, 55
   10 IF (FREQ(I,L) .GT. MAX)  MAX = FREQ(I,L)
C*******
C*******   PRINT THE GRAPH
C*******
      NL = NUMLN(K)
      DO 20 I = 1, NL
      NOX = FREQ(I,L) * 100 / MAX + 1
      IVAL = LBND(K) + (I-1) * NUMPL(K)
      DVAL = IVAL / SF(K)
      WRITE (KW,105) IVAL,DVAL,FREQ(I,L),(LINE(IX),IX=1,NOX)
  105 FORMAT(I10,F10.5,I8,2X,102A1)
   20 CONTINUE
      WRITE (KW,110) LBND(K),LCT(L),UBND(K),UCT(L)
  110 FORMAT (//T20,'# UNDER ',I8,' = ',I6,
```

```
     @//T20,'# OVER  ',I8,' = ',I6)
C*******
C*******  ADD THE TOTALS FOR THIS LEVEL TO THE TOTALS FOR THE NEXT LEVEL
C*******  AND ZERO THE TOTALS FOR THIS LEVEL
C*******
      IF (L .GT. 4)  GO TO 40
      J = L + 2
      DO 30 I = 1, 55
      FREQ(I,J) = FREQ(I,J) + FREQ(I,L)
   30 FREQ(I,L) = 0
      UCT(J) = UCT(J) + UCT(L)
      LCT(J) = LCT(J) + LCT(L)
      UCT(L) = 0
      LCT(L) = 0
   40 CONTINUE
   50 CONTINUE
      RETURN
      END
```

### 3.3 FILL_JAN91.FOR

```
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c%%%%%
c%%%%% fill_jan91.for

c##### Revision of FILL_AUG90.FOR   by K. Tracey    January 1991
c#####   Reworked the code for handling bad/missing records:
c##### When good records are interspersed between bad and missing records,
c##### earlier versions of the code did not keep these yearhours in their
c##### correct order. In the earlier codes, all missing records were added
c##### at once; thus causing a good yearhour to be put out of sequence.
c##### Now the code has been modified to add the missing records before and
c##### after the good yearhours, keeping them in their correct order.
c%%%%%
c%%%%% revision of fill_jul88.for
c%%%%% This version will make three output files: travel time, temperature,
c%%%%% and pressure.  Each will be assigned the proper time according to
c%%%%% the particular model PIES, URI or Sea Data. The motivation for
c%%%%% separating the records arose when it was found that the different
c%%%%% model PIES' didn't sample identically.  The sampling relative to the
c%%%%% travel time is:
c%%%%%   URI-  temp  -115 sec      SD-  temp   773.750 sec
c%%%%%         press -115 sec           press 1645.625 sec
c%%%%%
c%%%%% here the time represents the period AFTER the center of the travel
c%%%%% time measurement (center of the burst of 24 pings) when, for a given
c%%%%% scan the, particular sensor is sampled.
c%%%%%
c%%%%% A namelist was added to the control file which specifies if the
c%%%%% instrument is a pressure instrument, and if so what model.  "card3"
c%%%%% has variables "pies" and "model". The first character of "pies" is
c%%%%% checked for a "Y" or "y".  The first character of model is checked
c%%%%% for a "S" or "s" to designate a SD from a URI model echo sounder.
c%%%%% The relative sample times above are correspondingly added to the
c%%%%% yearhour column of the inputed memod file and outputed to files
c%%%%% *.tmp, *.prs, and *.fill.
C%%%%%
C%%%%% additional i/o units:
C%%%%%    kw2 (UNIT 15) - pressure output file
C%%%%%    kw3 (UNIT 16) - temperature output file
c%%%%%
c%%%%% Changes will be mostly lower case, but some capiTalization.
c%%%%% more later, Fields 4-Aug-90


C*******
C*******                  FILL.JULY88.For
C*******
C******* ORIGINALLY WRITTEN BY J. GUNN JANUARY 1980
C******* REVISED BY K. TRACEY SINCE 1981
C******* CONVERTED FOR VAX  S. WOOD 1988
C*******
C*******      THE PURPOSE OF THIS PROGRAM IS TO SEARCH THROUGH THE IES
```

```
C******* RECORD TO MAKE SURE THAT THE TIMES ARE INCREMENTING CORRECTLY.
C******* THERE ARE TWO TYPES OF ERRORS IN THE TIME BASE: 1) THE DATA
C******* RECORD FROM A SAMPLING PERIOD IS MISSING, AND 2) THE RECORD IS
C******* THERE BUT THE TIME ASSOCIATED WITH IT IS INCORRECT.  IF A
C******* RECORD IS MISSING, A NEW ONE IS INSERTED WITH INTERPOLATED
C******* VALUES.
C*******       THE USER CAN EITHER PROCESS THE TOTAL DATASET OR SEARCH
C******* THROUGH A SMALLER PORTION BY SPECIFIYING NSTRAT AND NSTOP TO
C******* SELECT THE RECORDS.  IF A DATA GAP GREATER THAN MAXDLT IS
C******* ENCOUNTERED, THE PROGRAM HALTS.
C*******
c%%%%%%%%%%%%%%%%%%%%%%
c%%%%% logical unit numbers have been changed to avoid using
c%%%%% units 5 and 6.
c%%%%%%%%%%%%%%%%%%%%%%
C******* I/O UNITS:
C*******      KR  (UNIT 17) - CONTROL PARAMETERS
C*******      KW  (UNIT 18) - USERS OUTPUT LOG
C*******      KR1 (UNIT 19) - INPUT DATASET FROM MEMOD
C*******      KW1 (UNIT 20) - OUTPUT DATA FILE
C*******
C*********************************************************************
C*******
      CHARACTER*60 HEADR
character*3 model,press
logical pies
      INTEGER*4 RECIN,recadd, RECOUT, RDOFF, MAXBAD
      INTEGER*4 NALL,NADD,NBAD,KR,KW,KW1
      INTEGER*4 FLAG,NFLAG,LSTREC,LOKREC
      INTEGER*4 LINECT,LINPPG,LINADD
      integer*4 index(300), need(300)
      REAL*4 MAXDLT, DELTAT
      REAL*4 TT,PR,TP,AM,YRHR
      REAL*4 LOKTT,LOKPR,LOKTP,LOKAM,LOKYHR
      REAL*4 LSTTT,LSTPR,LSTTP,LSTAM,LSTYHR
      REAL*4 TTADD,PRADD,TPADD,AMADD,YHRADD
      REAL*4 DLTTT,DLTPR,DLTTP,DLTAM,DIFYHR
      REAL*4 SAVTT(300),SAVPR(300),SAVTP(300),SAVAM(300)
      real*4 SAVYHR(300)
      REAL*4 OKDLT, IESDLT
      real*4 goodyr(0:300),goodtt(0:300), goodpr(0:300), goodtp(0:300)
      real*4 midokyr
      PARAMETER (KR=17,KW=18,KR1=19,KW1=20,kw2=15,kw3=16)
      PARAMETER (LINPPG=54)
      NAMELIST/CARD1/ HEADR
      NAMELIST/CARD2/ NSTART,NSTOP,MAXDLT,DELTAT
namelist/card3/ press,model
      DATA NADD/0/,NBAD/0/,NALL/0/
      DATA RECIN/0/,RECOUT/0/,RECADD/0/
      DATA MAXBAD/300/,LINECT/55/,LINADD/0/
      DATA FLAG/-1/,pies/0/
C*****
C***** Open I/O units and files
C*****
```

```
      OPEN(UNIT=KR,STATUS='OLD', FORM='FORMATTED', READONLY)
      OPEN(UNIT=KW,STATUS='NEW',FORM='FORMATTED')
      OPEN(UNIT=KR1,STATUS='OLD',FORM='FORMATTED', READONLY)
      OPEN(UNIT=KW1,STATUS='NEW',FORM='FORMATTED')
C*****
C*****
C*****   READ THE CONTROL PARAMETERS AND WRITE TO LOG
C*****
      READ(KR,NML=CARD1)
      READ(KR,NML=CARD2)
      READ(KR,NML=CARD3)
if ((press(1:1).eq.'y').or.(press(1:1) .eq.'Y')) then
pies=.true.
open(unit=kw2,status='new',form='formatted')
open(unit=kw3,status='new',form='formatted')

if ((model(1:1).eq.'s').or.(model(1:1).eq.'S')) then
tmp_tcf= 1645.645/3600.
prs_tcf=773.750/3600.
else
tmp_tcf=-115./3600.
prs_tcf=-115./3600.
endif
endif
      WRITE(KW,410) HEADR
 410  FORMAT(//A60)
      WRITE(KW,415) NSTART,NSTOP,MAXDLT,DELTAT
 415  FORMAT(//5X,'NSTART =',I5,9X,'NSTOP =',I5,9X,'MAXDLT(HRS)=',
     @F10.4,5X,'DELTAT =',F10.4)
C*******
C*******  RESET PARAMETERS
C*******
      NGAP = MAXDLT/DELTAT + 0.5
      IF (NGAP .LE. 300) GO TO 5
      WRITE(KW,416)
 416  FORMAT(' **********  MAXDLT TOO BIG FOR ARRAYS **********'/
     @       ' RESET MAXDLT OR CHANGE DIMENSIONS'/
     @       ' RUN TERMINATED')
      STOP 416
    5 CONTINUE
      IF (MAXBAD .NE. NGAP) MAXBAD = NGAP
C*******
C******* SKIP OVER INITIAL RECORDS IF DESIRED.
C******* INCREMENT INPUT AND OUTPUT COUNTERS.
C*******
      I=NSTART
   10 IF(I .LE. 1) GO TO 20
      READ(KR1,420,END=80) TT,PR,TP,AM,YRHR
  420 FORMAT(5E15.7)
      RECIN=RECIN + 1
      WRITE(KW1,420) TT,YRHR
if (pies) then
write(kw2,420) pr,yrhr+prs_tcf
write(kw3,420) tp,yrhr+tmp_tcf
```

```
endif

      RECOUT = RECOUT+1
      I=I-1
      GO TO 10
C*******
C******* BEGIN PROCESSING BY READING NEXT DATA RECORD.
C******* ASSUME ITS YEARHOUR IS CORRECT.
C*******
   20 CONTINUE
      READ(KR1,420,END=80) TT,PR,TP,AM,YRHR
      RECIN=RECIN+1
      WRITE(KW,425) RECIN,YRHR
  425 FORMAT(//5X,'FIRST RECORD OF THE SERIES IS INPUT REC# = ',I5,
     @5X,'YRHR = ',F12.5////)
C*******
C******* MAIN PROCESSING LOOP
C******* WRITE RECORD IF TIME IS GOOD
C******* SAVE VALUES AS 'LAST OKAY'
C*******
   25 WRITE(KW1,420) TT,YRHR
if (pies) then
write(kw2,420) pr,yrhr+prs_tcf
write(kw3,420) tp,yrhr+tmp_tcf
endif

      RECOUT=RECOUT+1
      LOKYHR=YRHR
      LOKTT=TT
      LOKPR=PR
      LOKTP=TP
      LOKAM=AM
C*******
C******* SAVE THE MOST RECENTLY READ DATA VALUES
C******* CHECK FOR END OF PROCESSING
C*******
   30 CONTINUE
      IF(NBAD .GE. MAXBAD) GO TO 90
      IF(RECIN .GE. NSTOP) GO TO 70
      LSTYHR=YRHR
      LSTTT=TT
      LSTPR=PR
      LSTTP=TP
      LSTAM=AM
C*******
C******* READ NEXT DATA RECORD, CHECK FOR PROPER SEQUENCING
C*******
      READ(KR1,420,END=80) TT,PR,TP,AM,YRHR
      RECIN=RECIN+1
      IESDLT=YRHR-LSTYHR
C*******
C******* A)  THE SEQUENCING IS WRONG, SAVE THIS ONE AS BAD
C******* AND THEN GO GET THE NEXT RECORD
C*******
```

```
      IF(ABS(IESDLT - DELTAT) .GT. 0.1) GO TO 35
C*******
C******* B) THIS ONE IS OKAY, BUT THE PREVIOUS RECORDS WERE BAD
C******* SO WORK ON CLEANING UP THE DATASET
C*******
      IF(NBAD .NE. 0) GO TO 40
C*******
C******* C) THIS ONE IS OKAY, PREVIOUS WERE OKAY, SO GET NEXT RECORD
C*******
      GO TO 25
C*******
C******* RECORD IS OUT OF SEQUENCE - SAVE IT
C*******
   35 CONTINUE
      NBAD = NBAD+1
      SAVYHR(NBAD)=YRHR
      SAVTT(NBAD)=TT
      SAVPR(NBAD)=PR
      SAVTP(NBAD)=TP
      SAVAM(NBAD)=AM
      GO TO 30
C*******
C******* PROPER SEQUENCING HAS RESUMED, BUT PREVIOUS RECORDS
C******* WERE OUT OF ORDER.
C*******
   40 CONTINUE
      DLTYHR=LSTYHR-LOKYHR
      OKDLT=DLTYHR-NBAD*DELTAT
      IF(OKDLT .GT. MAXDLT) GO TO 90
      NADD=NADD+RDOFF(OKDLT/DELTAT)
      NBAD=NBAD-1
      NALL=NBAD+NADD
C*******
C******* WRITE TO LOG - OUT OF SEQUENCE RECORDS
C*******
      IF(LINECT .LT.LINPPG) GO TO 50
      WRITE(KW,430)
  430 FORMAT('1')
      WRITE(KW,432)
  432 FORMAT(//7X,'LAST GOOD SEQUENCING',8X,'SEQUENCING RESUMED',
     @6X,'# ADDED',5X,'RECORDS WITH YRHRS'/
     @9X,'RECIN',4X,'LSTOK YRHR',8X,'RECIN',5X,'LST YRHR',
     @6X,'RECORDS',6X,'OUT OF SEQUENCE')
      WRITE(KW,434)
  434 FORMAT('+',2(5X,'__________  __________'),2(5X,'_______'),
     @'___________'//)
      LINECT=0
   50 CONTINUE
      LOKREC=RECIN-NBAD-2
      LSTREC=RECIN-1
      IF(NBAD .EQ.0) GO TO 52
      WRITE(KW,435) LOKREC,LOKYHR,LSTREC,LSTYHR,NADD,(SAVYHR(I),
     @I=1,NBAD)
  435 FORMAT(1X,2(5X,I10,2X,F10.2),5X,I7,5X,5F10.2,10(/72X,5F10.2))
```

```
      IF(MOD(NBAD,5) .NE. 0) LINADD=1
      LINADD=LINADD+1
      LINECT=LINECT+LINADD
      LINADD=0
      GO TO 55
   52 CONTINUE
      WRITE(KW,435) LOKREC,LOKYHR,LSTREC,LSTYHR,NADD
      LINECT=LINECT+1
C*******
C******* IF MISSING RECORDS, determine if any of the "bad" records may
c ****** in fact be good. We will want to use them if possible, and add
c******* missing records before and/or after them as necessary
C*******
   55 IF (NADD .gt. 0) then
         nparts = 1
         index(1) = nbad
         midokyr = lokyhr
         midindex = 0
         nleft = nadd
         goodyr(0) = lokyhr
         goodtt(0) = loktt
         goodpr(0) = lokpr
         goodtp(0) = loktp

         do 56 k = 1,nbad
            if (savyhr(k) .gt. lokyhr .and. savyhr(k) .lt. lstyhr) then

c***** This may be a "good" yearhour.  First make sure that it has proper
c***** incrementation.
               irem = int( (savyhr(k) - lokyhr) / deltat)
               have = lokyhr + irem*deltat

c***** Yes this is a good yearhour, determine how many records must be added
c***** before this one.
               if ( abs(have - savyhr(k)) .lt. 0.05 ) then
                  diff = savyhr(k) - midokyr
                  nwant = rdoff(diff/deltat)
                  nhave = k - midindex
                  nowneed = nwant - nhave
                  if (nowneed .le. nleft) then
                     if (nowneed .lt. 0) then
                       if (nowneed + need(nparts-1) .eq. 0) then
                           nleft  = nleft + need(nparts-1)
                           nparts = nparts - 1
                           nowneed = 0
                       else
                           go to 56
                       end if
                     end if
                     need(nparts) =  nowneed
                     nleft = nleft - need(nparts)
                     index(nparts) = k
                     goodyr(nparts) = savyhr(k)
                     goodtt(nparts) = savtt(k)
```

```
                     goodpr(nparts) = savpr(k)
                     goodtp(nparts) = savtp(k)
                     midokyr = savyhr(k)
                     midindex = k
                     nparts = nparts + 1
                  end if
               end if
            end if
  56     continue
         need(nparts) = nleft
         goodyr(nparts) = lstyhr
         goodtt(nparts) = lsttt
         goodpr(nparts) = lstpr
         goodtp(nparts) = lsttp
      end if
C*******
C******* INTERPOLATE IF NECESSARY AND WRITE OUT ALL 'SAVED' RECORDS
C*******
   60 CONTINUE
c*****
c***** Case 1: No records missing, but some yearhours were bad.
c***** Fix Up: Adjust the yearhours to be correct; don't add any records.

      if (nadd .eq. 0) then
          do 61 ii = 1, nbad
             yhradd = ii*deltat + lokyhr
             write(kw1,420) savtt(ii),yhradd
             if (pies) then
          write(kw2,420) savpr(ii),yhradd+prs_tcf
          write(kw3,420) savtp(ii),yhradd+tmp_tcf
     endif
             recout =  recout  + 1
             nflag = nflag + 1
   61     continue

      else

c***** Case 2: Records must be added during one or more sub-zones,
c***** delineated by "good" yearhours.
c***** Fix up: Added records before the "good" records if needed.
c***** Interpolating travel time, pressure, and temperature.

          yhradd = lokyhr
          do 69 k = 1, nparts
             if (k .eq. 1) then
                lfst = 1
             else
                lfst = index(k-1) + 1
             end if
             if (k .eq. nparts) then
                last = nbad
             else
                last = index(k) - 1
             end if
```

```

c*****  Add missing records first.
             if (need(k) .ne. 0) then
                DLTyr= goodyr(k) - goodyr(k-1)
                DLTTT= goodtt(k) - goodtt(k-1)
                DLTPR= goodpr(k) - goodpr(k-1)
                DLTTP= goodtp(k) - goodtp(k-1)
                do 63  kadd = 1, need(k)
                   yhradd = deltat + yhradd
                   difyhr=kadd*deltat/dltyr
                   ttadd=difyhr*dlttt+loktt
                   pradd=difyhr*dltpr+lokpr
                   tpadd=difyhr*dlttp+loktp
                   write(kw1,420) ttadd,yhradd
              if (pies) then
      write(kw2,420) pradd,yhradd+prs_tcf
      write(kw3,420) tpadd,yhradd+tmp_tcf
           endif
                   nflag=nflag+1
                   recadd=recadd+1
                   nadd=nadd-1
  63            continue
             end if

c ***** Next right out any records with bad yearhours.

             do 65 l = lfst, last
                yhradd = deltat + yhradd
                write(kw1,420) savtt(l),yhradd
        if (pies) then
    write(kw2,420) savpr(l),yhradd+prs_tcf
    write(kw3,420) savtp(l),yhradd+tmp_tcf
        endif
                nflag=nflag+1
                recout=recout+1
65           continue

c ***** Finally write out the "good" record if it lies between bad ones.

             if (k .lt. nparts) then
                write(kw1,420) savtt(index(k)),savyhr(index(k))
        if (pies) then
   write(kw2,420) savpr(index(k)),savyhr(index(k))+prs_tcf
   write(kw3,420) savtp(index(k)),savyhr(index(k))+tmp_tcf
        endif
                recout = recout + 1
                yhradd = savyhr(index(k))
             end if
69        continue
      end if
c ***** All finished with these records.


      NADD=0
```

```
      NBAD=0
      WRITE(KW1,420) LSTTT,LSTYHR
if (pies) then
write(kw2,420) lstpr,lstyhr+prs_tcf
write(kw3,420) lsttp,lstyhr+tmp_tcf
endif

      RECOUT=RECOUT+1
      IF(RECOUT .GE. NSTOP) GO TO 70
      GO TO 25
C*******
C******* END OF PROCESSING  -  WRAP UP
C******* READ AND WRITE ANY REMAINING RECORDS
C*******
   70 CONTINUE
      READ(KR1,420,END=85) TT,PR,TP,AM,YRHR
      RECIN=RECIN+1
      WRITE(KW1,420) TT,YRHR
if (pies) then
write(kw2,420) pr,yrhr+prs_tcf
write(kw3,420) tp,yrhr+tmp_tcf
endif

      RECOUT=RECOUT+1
      GO TO 70
C*******
C******* UNEXPECTED END OF INPUT DATA
C*******
   80 CONTINUE
      IF(NBAD .NE. 0) GO TO 40
      WRITE(KW,440)
  440 FORMAT(//5X,'UNEXPECTED END OF FILE ENCOUNTERED BEFORE NSTOP',
     @' RECORDS WERE READ')
C*******
C******* NORMAL END OF PROCESSING - WRITE TO USERS LOG
C*******
   85 CONTINUE
      WRITE(KW,442)RECIN,YRHR
  442 FORMAT(///5X,'LST RECORD OF THE SERIES IS INPUT REC# = ',
     @I5,5X,'YRHR = ',F12.5)
   86 CONTINUE
      RECOUT=RECOUT+RECADD
      WRITE(KW,444) RECIN,RECADD,RECOUT,NFLAG
  444 FORMAT(//5X,'TOTAL RECORDS READ = ',T35,I10,
     @/5X,'TOTAL RECORDS ADDED = ',T35,I10,
     @/5X,'TOTAL RECORDS OUTPUT =',T35,I10,
     @/5X,'TOTAL FLAGGED YRHRS =',T35,I10)

      STOP
C*******
C******* TOO MANY OUT OF SEQUENCE RECORDS IN-A-ROW, TERMINATE RUN
C*******
   90 CONTINUE
      WRITE(KW,446) MAXBAD,RECIN
```

```
  446 FORMAT(//5X,'MORE THAN ',I4,' CONSECUTIVE OUT-OF-SEQUENCE',
     @' RECORDS ENCOUNTERED',
     @//5X,'LAST INPUT RECORD READ WAS RECIN = ',I5,
     @//5X,'RUN TERMINATED')
      STOP 999
      END
C*******
C****************************************************************************
C*******
      INTEGER FUNCTION RDOFF(REAL)
      NUMBER=IFIX(REAL)
      REST=REAL-NUMBER
      IF(REST .LT. 0.5) GO TO 110
      NUMBER=NUMBER+1
  110 RDOFF=NUMBER
      RETURN
      END
```

## 3.4 MEMOD_JUL89.FOR

```
C*******
C*******                 MEMOD_Jul89.For
c******* this version was modified 19-jul-1989 the modifcations are
c******* documented and are made in lower case.  The major modifications
c******* are the addition of another window called binwindow.  This is
c******* described in the comment statements in the routine of that name.
c******* Another change related to the 97% confidence window applied within
c******* ttmode.
c*******
C******* THIS PROGRAM IS DESIGNED TO TAKE GROUPS OF IES TRAVEL TIMES
C******* (TTA AND/OR TTB) AND COMPUTE THE MEDIAN OR MODAL VALUE
C*******
C*******   ORIGINALLY WRITTEN 1979 BY J. GUNN, BUT HAS BEEN REVISED AND
C*******   REWRITTEN SEVERAL TIMES SINCE THEN.
C*******
C*******   THIS PROGRAM AT THE PRESENT TIME DOES THE FOLLOWING:
C******* 1)  TT1 AND TT2:
C******* THE PROGRAM IS NOW SET UP TO ALLOW PROCESSING OF BOTH TT1 AND
C******* TT2 DURING THE SAME RUN.  ORIGIANLLY TT2 COULD ONLY BE
C******* PROCESSED BY THE MEDIAN METHOD.  NOW THE USER CAN SPECIFY
C******* EITHER METHOD IN THE CONTROL FILE.  IF S/R TTMEDN IS USED THE
C******* CALCULATIONS ARE DONE AS INTEGERS THEN PASSED BACK TO THE
C******* CALLING PROGRAM AS REALS.  IF OVERRANGING HAS TAKEN PLACE,
C******* WRAPPING WILL BE DONE AUTOMATICALLY AS LONG AS THE UBND IS
C******* SPECIFIED SMALLER THAN THE LBND.  THE OUTPUT IS A SINGLE
C******* TT FOR A GIVEN SAMPLING PERIOD - SCLAED TO SECONDS.
C*******
C******* 2)  PRESSURE AND TEMPERATURE
C******* ASSUMES THE SENSORS ARE PAROS INSTRUMENTS.  THE USER SUPPLIES
C******* THE COEFFICIENTS FOR PRESSURE AND CALIBRATION VALUES FOR
C******* TEMPERATURE.  TWO PAROS EQUATIONS CAN BE USED EITHER THE A,B
C******* OR THE C,D EQUATION.  THE TEMPERATURE DEPENDENT COEF. ARE
C******* CALCULATED USING TEMP (F) UNLESS TWO OF THE D ONES ARE
C******* EQUAL.  THEN TEMP(C) IS USED.  OVERRANGING IS TAKEN INTO
C******* ACCOUNT IF THE USER SPECIFIES IT.  ON OUTPUT PRESSURE AND
C******* TEMPERATURE COUNTS ARE CONVERTED TO DBAR AND T(C).  IF NEITHER
C******* SENSOR WAS USED, THE OUTPUT VALUES ARE -99.
C*******
C******* 3) TIME BASE
C******* TIME BASE IS SET RELATIVE TO ISEQO, SUPPLIED BY THE USER.
C******* ASSUMES THAT INPUT TIME IS ALREADY GMT.  ON OUTPUT, ALL
C******* SEQUENCE NUMBER ARE CONVERTED TO TIME IN YEARHOURS.  THESE
C******* CAN BE POSITIVE OR NEAGATIVE, DEPENDING ON ISEQO.
C*******
C******* INPUT/OUTPUT UNITS USED:
C******* KR      (UNIT   5) - CONTROL INPUT
C******* KW      (UNIT   6) - LOG OUTPUT
C******* KWDA (UNIT   7) - TT1 MODE/MEDIAN DISK OUTPUT DATASET
C******* KWDB (UNIT   8) - TT2 MODE/MEDIAN DISK OUTPUT DATASET
C******* KWLA (UNIT   9) - TT1 LISTING OF STATISTICS
C******* KWLB (UNIT  10) - TT2 LISTING OF STATISTICS
C******* KRBUNS  (UNIT  11) - INTEGER INPUT OF BUNS DATA
```

```
C*******
      CHARACTER*60 HEADR
      CHARACTER*4 IOPT(6), MED1, MED2, MOD1, MOD2, TT1, TT2
      CHARACTER*3 TTYPE(2)
      CHARACTER*2 SENSOR(3), PR, TP, AM
      CHARACTER*2 EQN, TYPEQN(2), OVERNG, YES
C*******
      INTEGER*4 KR, KW, KWDA, KWDB, KWLA, KWLB
      INTEGER*4 NTT
      INTEGER*4 NWORDS, LBURST, LBFST, RDFMT
      INTEGER*4 NSEN, SWDNO(3)
      INTEGER*4 NFIRST, NFSEQ, NLAST, NLSEQ, SEQINC
      INTEGER*4 LBND1, UBND1, LBND2, UBND2
      INTEGER*4 LAB, CTREF1, CTREF2
      INTEGER*4 IX(100), IXB(100), IXX(100), IARRAY(100)
      INTEGER*4 PWR2, PMID, TWOPWR(17)
      INTEGER*4 PRWDNO, TPWDNO, AMWDNO
      INTEGER*4 RECIN, SEQNO, ZERO, FOUR
      INTEGER*4 IPASS, ISEQO, IPRESS, ITEMP, IAMBNS
C*******
      INTEGER*2 PRCES1, PRCES2, TT1SW, TT2SW
      INTEGER*2 PRSN, TPSN, AMSN
      INTEGER*2 LINCTR,PGCTR,LINPPG
      INTEGER*2 ENDFLG, OLD, NEW
C*******
      REAL*4 XMED,RANGE1,RANGE2,AMSF,SF1,SF2
      REAL*4 ARRAY(100),X1,X2,GRPHR
      REAL*4 PRESS,TEMP,AMBNS,MTIME
      REAL*8 GYRHR,DTIME,DGRPHR,OFFSET
      REAL*8 AC1,AC2,AC3,BD1,BD2,BD3,T1,T2,T3,T4
C*******
      PARAMETER(TT1=' TT1', TT2=' TT2')
      PARAMETER (MED1 ='MED1', MED2 = 'MED2', MOD1='MOD1', MOD2='MOD2')
      PARAMETER(PR='PR',TP='TP',AM='AM')
      PARAMETER(LINPPG=54, YES = 'YE')
      PARAMETER(KR=5, KW=6, KWDA=7, KWDB=8, KWLA=9, KWLB=10)
C*******
      COMMON/COMMED/ IARRAY
      COMMON/COMMOD/ ARRAY
      COMMON/MEMOCM/XMED,NGOODP,SDQRT,KT
      COMMON/PARAM/NSKIP,NFIRST,LBURST
      COMMON/UNIT/KRBUNS,RDFMT
      COMMON/PCOEF/AC1,AC2,AC3,BD1,BD2,BD3,T1,T2,T3,T4
      COMMON/TCOEF/TREF1,TREF2,CTREF1,CTREF2,TSEC,LAB
      COMMON/PRSEQN/ EQN,OVERNG
      COMMON/INDX/KF1,KL1,KF2,KL2
C*******
      NAMELIST/CARD1/ HEADR
      NAMELIST/CARD2/ NTT, TTYPE
      NAMELIST/CARD3/ NWORDS, LBURST, LBFST, RDFMT
      NAMELIST/CARD4/ NSEN, SENSOR, SWDNO
      NAMELIST/CARD5/ SF1, SF2, AMSF
      NAMELIST/CARD6/ NFIRST, NFSEQ, NLAST, NLSEQ, SEQINC
      NAMELIST/CARD7/ LBND1, UBND1, LBND2, UBND2, DGRPHR
```

```
      NAMELIST/CARD8/ IOPT
      NAMELIST/CARD9/ IYR, MNTH, IDAY, IHOUR, MINUT, ISEC, ISEQO
      NAMELIST/CARD10/ EQN,OVERNG
      NAMELIST/CARD11/ AC1,AC2,AC3
      NAMELIST/CARD12/ BD1,BD2,BD3
      NAMELIST/CARD13/ T1,T2,T3,T4
      NAMELIST/CARD14/ LAB,TSEC,TREF1,TREF2,CTREF1,CTREF2
C*******
      DATA KRBUNS/11/
      DATA TYPEQN/'AB','CD'/
      DATA LINCTR/60/,PGCTR/0/
      DATA TT1SW/0/,TT2SW/0/,PRCES1/0/,PRCES2/0/
      DATA ENDFLG/0/, OFFSET/0.0/,IPASS/0/
      DATA SF1/20480.0/,SF2/20480.0/,ZERO/0/,FOUR/4/
      DATA RECIN/0/,IOPT/6*'    '/
      DATA PRESS/-99./,TEMP/-99./,AMBNS/-99./
      DATA SENSOR/3*'  '/,SWDNO/0,0,0/
      DATA PRSN/0/,TPSN/0/,AMSN/0/
      DATA TREF1/0./,TREF2/0./,CTREF1/0/,CTREF2/0/
      DATA LAB/0/,TSEC/0.0/, MIDOPT/0/
      DATA TWOPWR/2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,
     @            16384,32768,65536,131072/
C*****
C*****
      write(*,42)
 42   format(1X,//'  PROGRAM IS RUNNING. PLEASE WAIT.',//,
     @'        HAVE SOME JAVA!!',//)
C*******  OPEN THE CONTROL CARD FILE AND LOG FILE.
C*******  THE I/O FILES WILL BE OPENED LATER AS NEEDED
C*******
      OPEN(UNIT=KR, STATUS='OLD', FORM='FORMATTED', READONLY)
      OPEN(UNIT=KW, STATUS='NEW', FORM='FORMATTED')
C*******
C*******  READ THE CONTROL PARAMETERS FOR TYPES OF IES
C*******
      READ(KR,NML=CARD1)
      READ(KR,NML=CARD2)
      READ(KR,NML=CARD3)
      READ(KR,NML=CARD4)
      READ(KR,NML=CARD5)
      READ(KR,NML=CARD6)
      READ(KR,NML=CARD7)
      READ(KR,NML=CARD8)
      READ(KR,NML=CARD9)
C*******
C******* READ IN CONTROL PARAMETERS FOR THOSE WITH ADDITIONAL SENSORS
C*******
      IF(NSEN .NE. 0) THEN
         READ(KR,NML=CARD10)
         READ(KR,NML=CARD11)
         READ(KR,NML=CARD12)
         READ(KR,NML=CARD13)
         READ(KR,NML=CARD14)
      END IF
```

```
C*******
C******* CHECK FOR OVERRANGING OF EITHER TT1 OR TT2
C******* ASSUMES THAT ONLY ONE WILL OVERRANGE DURING A DEPLOYMENT.
C******* RESETS THE UPPER BOUND IF NECESSARY.
C*******
   45 IF(UBND1 .LT. LBND1) THEN
         MIDOPT=1
         DO 500 LP=1,17
         NOW2=TWOPWR(LP)
         IF(NOW2 .GE. LBND1) THEN
            PWR2=NOW2
            PMID=TWOPWR(LP-1)
            UBND1=UBND1+PWR2
            GO TO 505
         END IF
  500    CONTINUE
      END IF
  505 CONTINUE
      IF(UBND2 .LT. LBND2) THEN
         MIDOPT=2
         DO 510 LP=1,17
         NOW2=TWOPWR(LP)
         IF(NOW2 .GE. LBND2) THEN
            PWR2=NOW2
            PMID=TWOPWR(LP-1)
            UBND2=UBND2+PWR2
            GO TO 525
         END IF
  510    CONTINUE
      END IF
  525 CONTINUE
C*******
C*******   WRITE OUT THE CONTROL PARAMETERS
C*******
      WRITE(KW,310) HEADR
  310 FORMAT(T20,' ***** MEMOD PROGRAM OUTPUT *****'/T10,A60//)
      WRITE(KW,315) NTT,(TTYPE(I),I=1,2),NSEN,(SENSOR(I),I=1,3)
  315 FORMAT(' THE FOLLOWING SENSORS ARE AVAILABLE: '/
     @ I10,' TRAVEL TIME DETECTORS: ',2A4/
     @ I10,' ADDITIONAL SENSORS: ',3(A2,2X)//)
      WRITE(KW,320) NFIRST,NFSEQ,NLAST,NLSEQ,DGRPHR,SEQINC,LBURST
  320 FORMAT(
     @' RECORD #''S',I8,' (SEQ # ',I8,') THRU ',I8,'(SEQ #',I8,')',
     @' WERE PROCESSED'//' SAMPLING RATE IS ',D15.9,
     @' DATA GROUPS PER HOUR(SEQINC = ',I5,')'//
     @' ',I10,' DATA POINTS WERE USED FOR EACH MEDIAN VALUE'///)
      WRITE(KW,330)LBND1,UBND1,LBND2,UBND2,(IOPT(I),I=1,6)
  330 FORMAT(' PROCESSING PARAMETERS:'//
     @ ' ',4X,'TT1MIN',4X,'TT1MAX',4X,'TT2MIN',4X,'TT2MAX'/' ',4I10//
     @ ' OPTIONS IN EFFECT = ',6(2X,A4)/
     @ 10X,' NOTE: TT2 IS ONLY PROCESSED BY MEDIAN'//)
      WRITE(KW,332) SF1,SF2,AMSF
  332 FORMAT(' SCALING FACTORS FOR TT1, TT2, AND AMBNS ARE: ',
     @3F10.2//)
```

```
      IF(MIDOPT .NE. 0) WRITE(KW,335) MIDOPT,PWR2,PMID
  335 FORMAT(' TT DETECTOR #',I2,' WAS FOLDED TO AVOID',
     @ ' WRAP-AROUND'/
     @ 1X,I10,' WAS ADDED TO ALL POINTS LESS THAN', I10//)
      IF(NSEN .EQ. 0) GO TO 48
      IF(EQN .EQ. TYPEQN(1)) WRITE(KW,344)
  344 FORMAT(' PAROS EQUATION USED:  P = A(1-TO/T) - B(1-TO/T)**2')
      IF(EQN .EQ. TYPEQN(2)) WRITE(KW,346)
  346 FORMAT(' PAROS EQUATION USED:  P = C{[1 - (TO/T)**2] - ',
     @        'D[1 - (TO/T)**2]**2}')
      IF(OVERNG .EQ. YES) WRITE(KW,348)
  348 FORMAT(' PRESSURE OVER-RANGED AT DEPTH 2**24 ADDED TO THE COUNTS')
      WRITE(KW,341) AC1,AC2,AC3,BD1,BD2,BD3,T1,T2,T3,T4
  341 FORMAT(' PRESSURE COEFFICIENTS:'//
     @6X,'AC1',11X,'AC2',11X,'AC3'/3(1X,D12.5,1X)/
     @6X,'BD1',11X,'BD2',11X,'BD3'/3(1X,D12.5,1X)/
     @6X,'T1',12X,'T2',12X,'T3',12X,'T4'/4(1X,D12.5,1X)//)
      WRITE(KW,342) TSEC,TREF1,TREF2,CTREF1,CTREF2
  342 FORMAT(' SAMPLING TIME (SEC) FOR PRESS AND TEMP IS',F10.5//
     @' CALIBRATION TEMPERATURES: ',2F10.5/
     @'                   COUNTS: ',2I10)
C*******
C*******  SET OPTION SWITCHES
C*******
   48 DO 50 I=1,6
      IF (IOPT(I) .EQ. MED1) THEN
         PRCES1=1
      ELSE IF (IOPT(I) .EQ. MED2) THEN
         PRCES2=1
      ELSE IF (IOPT(I) .EQ. MOD1) THEN
         PRCES1=0
      ELSE IF (IOPT(I) .EQ. MOD2) THEN
         PRCES2=0
      ELSE IF (IOPT(I) .EQ. TT2) THEN
         TT2SW=1
         OPEN(UNIT=KWDB, STATUS='NEW', FORM='FORMATTED')
         OPEN(UNIT=KWLB, STATUS='NEW', FORM='FORMATTED')
      ELSE IF (IOPT(I) .EQ. TT1) THEN
         TT1SW=1
         OPEN(UNIT=KWDA, STATUS='NEW', FORM='FORMATTED')
         OPEN(UNIT=KWLA, STATUS='NEW', FORM='FORMATTED')
      END IF
   50 CONTINUE
      IF(NSEN .NE. 0) THEN
      DO 55 I = 1, 3
         IF (SENSOR(I) .EQ. PR) THEN
            PRSN = 1
            PRWDNO = SWDNO(I)
   53    ELSE IF (SENSOR(I) .EQ. TP) THEN
            TPSN = 1
            TPWDNO = SWDNO(I)
   54    ELSE IF (SENSOR(I) .EQ. AM) THEN
            AMSN = 1
            AMWDNO = SWDNO(I)
```

```
          END IF
   55     CONTINUE
       END IF
C*******
C******* OPEN THE BUNS INPUT DATA SET DEPENDING ON THE FORMAT
C*******
       IF (RDFMT .EQ. 0) THEN
           OPEN(UNIT=KRBUNS, STATUS='OLD', FORM='UNFORMATTED')
       ELSE
           OPEN(UNIT=KRBUNS, STATUS='OLD', FORM='FORMATTED')
       END IF
C*******
C******* CALCULATE THE TIME BASE:
C******* TIME IS REFERENCED TO SEQUENCE NUMBER ISEQO, ASSUMES THAT
C******* TIME IS GIVEN AS GMT.  THE MEDIAN TIME ASSOCIATED WITH
C******* ALL RECORDS IS OFFSET TO PLACE IT IN THE MIDDLE OF THE SAMPLE
C******* BURST
C******* WRITE OUT TIME BASE TO LOG.
C*******
   65 CONTINUE
       CALL YRDAY(IYR,MNTH,IDAY,IYRDAY)
       CALL GMTYR(IYRDAY,IHOUR,MINUT,ISEC,GYRHR,ZERO,FOUR)
       OFFSET=DFLOAT(LBURST-1)*10.0/7200.0
       GYRHR = GYRHR + OFFSET
       WRITE(KW,354) ISEQO,IYR,MNTH,IDAY,IHOUR,MINUT,ISEC
  354 FORMAT(//' TIME BASE PARAMETERS:'/
     @' SEQUENCE NUMBER',I10,' IS ASSIGNED THE FOLLOWING TIME:'/
     @8X,'IYR',6X,'MNTH',6X,'IDAY',5X,'IHOUR',5X,'MINUT',6X,'ISEC',
     @/1X,6I10)
       WRITE(KW,355) GYRHR,ISEQO
  355 FORMAT(/' GYRHR =',D15.9,' FOR ISEQO =',I10//)
C*******
C******* CALCULATE TOTAL NUMBER OF ECHOS AND RANGES OF THEM
C*******
       KOUNT = LBURST * NTT
       IF (TT1SW .NE. 0) THEN
          RN1MIN = FLOAT(LBND1)
          RN1MAX = FLOAT(UBND1)
          RANGE1 = (RN1MAX - RN1MIN) /SF1
       END IF
       IF (TT2SW .NE. 0) THEN
          RN2MIN = FLOAT(LBND2)
          RN2MAX = FLOAT(LBND2)
          RANGE2 = (RN2MAX - RN2MIN) /SF2
       END IF
C*******
C*******     ENTER MAIN PROCESSING LOOP
C*******
 6666 CONTINUE
C*******
C******* WRITE PAGE HEADING TO LOG IF NEEDED
C******* TYPE OF HEADER DEPENDS ON TT DETECTOR TYPE
C*******
       IF (LINCTR .GE. LINPPG) THEN
```

```
         LINCTR=0
         PGCTR = PGCTR + 1
  600    IF (TT1SW .NE. 0) THEN
            IF (PRCES1 .NE. 1) THEN
               WRITE(KWLA,410) HEADR,PGCTR
            ELSE
  602          WRITE(KWLA,400) HEADR,PGCTR
            END IF
  604       WRITE(KWLA,415)
         END IF
  605    IF (TT2SW .NE. 0) THEN
            IF (PRCES2 .NE. 1) THEN
               WRITE(KWLB,410) HEADR, PGCTR
            ELSE
  606          WRITE(KWLB,415) HEADR,PGCTR
            END IF
  608       WRITE(KWLB,415)
         END IF
      END IF
  400    FORMAT(1H1,T10,A60,T110,'PAGE',I4,//T8,'SEQ#',T20,'MEDIAN',
     @   T32,'QUARTILE',T46,'#GOOD',T61,'XO',T69,'PRESS DBAR',
     @   T83,'TEMP (C)',T96,'AMB NOISE',T109,'TIME(MID)')
  410    FORMAT(1H1,T10,A60,T110,'PAGE',I4,//T8,'SEQ#',T19,'  MODE',
     @   T35,'SD',T46,'#GOOD',T61,'XO',T69,'PRESS DBAR',
     @   T83,'TEMP (C)',T96,'AMB NOISE',T109,'TIME(MID)')
  415    FORMAT('+  ',9('__________',3X))
C*******
C*******   READ IN THE DATA.
C*******   INCREMENT COUNTER AND SET SEQNO
C*******
  610 CONTINUE
      CALL RDBUNS(NWORDS,IX)
      RECIN = RECIN + 1
      SEQNO=IX(1)
C*******
C******* SKIP UNWANTED RECORDS.  IF FIRST ONE TO PROCESS,
C******* MAKE SURE THE SEQUENCE NUMBER IS THE ONE EXPECTED.
C*******
      IF (IPASS .LE. 0) THEN
         IF(RECIN .LT. NFIRST) GO TO 610
         IF(SEQNO .EQ. NFSEQ) GO TO 625
         WRITE(KW,358) RECIN,SEQNO,NFSEQ
  358    FORMAT(5X,'*****   RUN TERMINATED   *****',
     @   /' FOR RECORD # ',I8,' FIRST SEQNO WAS ',I8,' INSTEAD OF ',I8)
         STOP 358
      END IF
C*******
C*******  CHECK FOR LAST RECORD TO BE PROCESSED
C*******  WRITE A WARNING IF RECIN GETS BIGGER THAN EXPECTED
C*******
  616 CONTINUE

      IF (RECIN .EQ. NLAST) THEN
         ENDFLG = 1
```

```
      ELSE IF (RECIN .GE. NLAST) THEN
  620    WRITE(KW,362)RECIN,NLAST,SEQNO
  362    FORMAT(' WARNING: NLAST EXCEEDED!  CHECK SEQUENCE NUMBERS. '/
     @    ' RECIN = ',I10,' NLAST = ',I10,5X,' SEQNO = ',I10/
     @    ' THIS GROUP HAS BEEN INCLUDED IN THE DATA SET'/
     @    ' PROGRAM TERMINATES NORMALLY.'//)
         ENDFLG = 1
      END IF
C*******
C******* CHECK FOR END OF FILE FLAG.  OTHERWISE, THIS RECORD
C******* IS TO BE PROCESSED, RENAME THE VARIABLES
C*******
  625 CONTINUE
      IF(SEQNO .EQ. -1) GO TO 7000
      IF(PRSN .EQ. 1) IPRESS = IX(PRWDNO)
      IF(TPSN .EQ. 1) ITEMP  = IX(TPWDNO)
      IF(AMSN .EQ. 1) IAMBNS = IX(AMWDNO)
      DO 626 K=1,KOUNT
      IXB(K)=IX(K+LBFST-1)
  626 CONTINUE
C*******
C******* ASSIGN THE TIME TO THE MIDDLE OF THE GROUP INTERVAL
C*******
      DTIME = GYRHR + (DFLOAT(SEQNO - ISEQO)/DGRPHR)/DFLOAT(SEQINC)
      MTIME = DTIME
C*******
C*******  WRITE OUT COUNTS OF FIRST SAMPLING PERIOD TO LOG FILE
C*******
      IF(SEQNO .EQ. NFSEQ) WRITE(KW,360)SEQNO,DTIME,
     @ (IXB(J),J=1,KOUNT)
  360 FORMAT(//1H1,4X,'FIRST POINTS READ:'//'  SEQNO =',I10,
     @ '  DTIME = ',E15.9/(4(2X,2I10)))
C*******
C*******  CALCULATE THE REAL PRESSURE AND TEMPERATURE, IF NECESSARY
C*******
      IF(PRSN .EQ. 1)  CALL TEMPRS(IPRESS,ITEMP,PRESS,TEMP)
C*******
C*******  CALCULATE THE AMBIENT NOISE, IF NECESSARY
C*******
      IF(AMSN .EQ. 1) AMBNS = IAMBNS/AMSF
C*******
C******* IF OVERRANGING, WRAP EITHER TT1 OR TT2.  WILL NOT DO BOTH
C*******
      IF (MIDOPT .NE. 0) THEN
         DO 630 J=MIDOPT,KOUNT,NTT
         IF (IXB(J) .LT. PMID) IXB(J) = IXB(J) + PWR2
  630    CONTINUE
      END IF
  635 CONTINUE
C*******
C*******  ORDER THE DATA
C*******
      CALL RESORT(KOUNT,NTT,IXB,IXX)
C*******
```

```
C******* PROCESS TT1 IF DESIRED
C******* IF 2 TT DETECTORS USED, PASS ONLY THE FIRST HALF OF IXX
C******* OTHERWISE ALL COUNTS WILL BE PASSED.
C******* CHECK WHETHER TO USE MODE OR MEDIAN PROCESS
C*******
      IF (PRCES1 .EQ. 0) THEN
         KT = 0
         DO 640 K=KF1, KL1
           KT = KT + 1
           ARRAY(KT) = IXX(K)
  640    CONTINUE
         CALL TTMODE(SEQNO,RN1MIN,RN1MAX)
      ELSE
  642    CONTINUE
         KT = 0
         DO 644 K=KF1, KL1
           KT = KT + 1
           IARRAY(KT) = IXX(K)
  644    CONTINUE
         CALL TTMEDN(LBND1,UBND1)
      END IF
C*******
C*******  SCALE THE COUNTS TO SECONDS AND WRITE TO DISK AND LOG
C*******
 646  X1 = XMED/SF1
      WRITE(KWDA,420) X1,PRESS,TEMP,AMBNS,MTIME
  420 FORMAT(5(2X,E13.7))
      WRITE(KWLA,425)SEQNO,XMED,SDQRT,NGOODP,X1,PRESS,TEMP,AMBNS,MTIME
  425 FORMAT(3X,I10,2(3X,F10.2),3X,I10,5(2X,F11.4))
C*******
C*******  PROCESS TT2 IF DESIRED
C*******    IF 2 TT DETECTORS PASS ONLY THE SECOND HALF OF IXX
C*******
  650 IF (TT2SW .NE. 0) THEN
         IF(PRCES2 .EQ. 0) THEN
            KT = 0
            DO 652 K=KF2,KL2
              KT = KT + 1
              ARRAY(KT) = IXX(K)
  652       CONTINUE
            CALL TTMODE(SEQNO,RN2MIN,RN2MAX)
         ELSE
            KT = 0
            DO 656 K=KF2,KL2
              KT = KT + 1
              IARRAY(KT) = IXX(K)
  656       CONTINUE
            CALL TTMEDN(LBND2,UBND2)
         END IF
C*******
C*******  SCALE FROM COUNTS TO SECONDS, AND WRITE TO DISK AND LOG
C*******
         X2 = XMED/SF2
         WRITE(KWDB,420)X2,PRESS,TEMP,AMBNS,MTIME
```

```
         WRITE(KWLB,425)SEQNO,XMED,SDQRT,NGOODP,X2,PRESS,TEMP,AMBNS,MTIME
      END IF
C*******
C*******   END OF MAIN PROCESSING LOOP
C*******   INCREMENT THE COUNTERS, CHECK FOR END OF DATA
C*******
      IPASS=IPASS+1
      LINCTR = LINCTR + 1
      IF (ENDFLG .EQ. 1) GO TO 7100
      GO TO 6666
C*******
C******* UNEXPECTED END OF DATA TERMINATE PROGRAM
C*******
 7000 CONTINUE
      WRITE(KW,370)
  370 FORMAT(//5X,' UNEXPECTED END OF DATA - RUN TERMINATED')
C*******
C******* NORMAL END OF PROCESSING WRITE MESSAGE TO LOG
C*******
 7100 CONTINUE
      WRITE(KW,375) SEQNO,DTIME,(IXB(J),J=1,KOUNT)
  375 FORMAT(//1H0,4X,'LAST POINTS READ:'//'  SEQNO =',I10,
     @ '  DTIME = ',E15.9/(4(2X,2I10)))
C*******
C******* WRITE TT1 PROCESSING MESSAGES
C*******
      IF (TT1SW .NE. 0) THEN
            IF (PRCES1 .EQ. 0) THEN
 7125          WRITE(KW,390) IPASS,TTYPE(1),RANGE1
            ELSE
               WRITE(KW,380) IPASS,TTYPE(1),RANGE1
            END IF
      END IF
C*******
C******* WRITE TT2 PROCESSING MESSAGES
C*******
 7130 IF (TT2SW .NE. 0) THEN
         IF(PRCES2 .EQ. 0) THEN
 7132       WRITE(KW,390) IPASS,TTYPE(2),RANGE2
         ELSE
            WRITE(KW,380) IPASS,TTYPE(2),RANGE2
         END IF
      END IF
  380 FORMAT(/5X,I10,' RECORDS WERE WRITTEN TO THE',A4,' MEDIAN DATASET
     @'/  15X,'RANGE OF DATA IS ',F10.5,' SEC')
  390 FORMAT(/5X,I10,' RECORDS WERE WRITTEN TO THE',A4,' MODAL DATASET
     @'/  15X,'RANGE OF DATA IS ',F10.5,' SEC')
  395 FORMAT(/5X,I10,' RECORDS WERE WRITTEN TO THE',A4,' MEDIAN DATASET
     @'/  15X,'RANGE OF DATA IS ',F10.5,' SEC')
      WRITE(*,43)
 43   FORMAT(1X,//,'  AH HA!! NOW YOU GOT TO DO SOME WORK!!'
     @,//,'  I AM FINISHED - SO HAVE FUN!! ',//)
      STOP
      END
```

```
C*******
C*********************************************************************
C*******
C*******                       SUBROUTINES
C*******
C*********************************************************************
C*******
      SUBROUTINE RDBUNS(NWORDS,IN)
      COMMON/UNIT/KRBUNS,RDFMT
      INTEGER*4 IN(100),RDFMT
C*******
C*******   READ EITHER BINARY OR FORMMATTED DATA
C*******
      IF(RDFMT .EQ. 0) THEN
         READ(KRBUNS,END=99)(IN(I),I=1,NWORDS)
      ELSE
   90    READ(KRBUNS,89,END=99) (IN(I),I=1,NWORDS)
   89    FORMAT(8I10)
      END IF
      RETURN
   99 IN(1)=-1
  100 CONTINUE
      RETURN
      END
C*******
C**********************************************************************
C*******
      SUBROUTINE RESORT(KOUNT,NTT,XB,OUT)
C*******
C******* NEW VERSION OF THE S/R ORDER
C******* THIS PUTS ALL THE COUNTS FROM ONE DETECTOR NEXT TO EACH OTHER
C******* AND PUTS THE COUNTS FROM THE SECOND ONE AFTER THOSE OF THE
C******* FIRST.  ONLY THEN ARE THE COUNTS FROM EACH DETECTOR SORTED
C******* FROM LOW TO HIGH
C*******
      COMMON/INDX/ KF1,KL1,KF2,KL2
      INTEGER*4 OUT(226),TEMP,XB(226)
C*******
C******* COPY XB TO OUT.  KEEP TRACK OF INDEX OF FIRST AND LAST
C******* TT FROM BOTH DETECTORS.
C*******
      K = 0
      DO 20 I=1,NTT
      DO 10 J=I,KOUNT,NTT
      K=K+1
      OUT(K) = XB(J)
   10 CONTINUE
      KL1 = K
      KF2 = K+1
   20 CONTINUE
      KF1 = 1
      KL2 = KOUNT
C*******
C*******  SORT THE MEMBERS OF FIRST TT DETECTOR FIRST.
```

```
C*******  THEN SORT THE SECOND TT MEMBERS.
C*******
      LAST = KOUNT - NTT
      DO 60 IN =1,NTT
      IF (IN .GT. 1) GO TO 25
      IFST = KF1
      ILST = KL1
      GO TO 26
   25 CONTINUE
      IFST = KF2
      ILST = KL2
c******
c****** test for removing no echos
c******
26 do 59  i=1,24
if ((out(i).eq.4351).or.(out(i).eq.4352)) out(i)=-out(i)
59 continue

      DO 50 I = IFST, ILST
      IF (OUT(I) .LE. OUT(I+1))  GO TO 50
      J = I + 1
      TEMP = OUT(J)
  30  K = J - 1
      IF (OUT(K) .LE. TEMP)  GO TO 40
      OUT(J) = OUT(K)
      J = J - 1
      IF (J .GT. IN)  GO TO 30
  40  OUT(J) = TEMP
  50  CONTINUE
  60  CONTINUE
      RETURN
      END
C*******
C***********************************************************************
C*******
      SUBROUTINE TTMEDN(RNMIN,RNMAX)
C*******          TAKES MEDIAN OF ARRAY XX.
C******* FIRST WINDOWS ARRAY WITHIN RANGE = (RNMIN,RNMAX)
C******* RETURNS NGOODP (# GOOD POINTS ) WITHIN RANGE
C******* AND CALCULATES MEDIAN AND QUARTILE RANGE OF THOSE POINTS
C******* ORIGINALLY WRITTEN SEPT 1978, A. CUTTING
C******* REWRITTEN BY K. TRACEY JULY 1985: NOW ALL CALCULATIONS
C******* ARE DONE AS INTEGERS, THEN PASSED BACK AS REALS.
C*******
      COMMON/COMMED/ XX
      COMMON /MEMOCM/ XMED, NGOODP, QUART, LBURST
      INTEGER*4 RNMIN,RNMAX, XX(100)
C*******
C*******  ELIMINATE THE OUT OF RANGE RAW DATA POINTS
C*******
      IBOT = 1
   42 IF (XX(IBOT) .GE. RNMIN)  GO TO 44
      IBOT = IBOT + 1
      IF (IBOT .LE. LBURST) GO TO 42
```

```
      XMED = RNMIN
      GO TO 47
C*******
C*******
C*******
   44 CONTINUE
      ITOP = LBURST
   46 IF (XX(ITOP) .LE. RNMAX)  GO TO 48
      ITOP = ITOP - 1
      IF (ITOP .GT. IBOT)  GO TO 46
      XMED = RNMAX
C*******
C*******  ALL POINTS IN THE INTERVAL ARE OUT OF RANGE
C*******
   47 CONTINUE
      NGOODP = 0
      QUART = 0.0
      RETURN
C*******
C*******  COMPUTE THE MEDIAN AND THE QUARTILE OF THE GOOD POINTS
C*******
   48 CONTINUE
      NGOODP = ITOP - IBOT + 1
      INDMED = IBOT + NGOODP / 2
      INDQLO = IBOT + (NGOODP + 2) / 4
      INDQHI = INDQLO + (NGOODP - 1) / 2
      XMED = XX(INDMED)
      QUART = XX(INDQHI) - XX(INDQLO)
   49 CONTINUE
      RETURN
      END
C*******
C***************************************************************************
C*******
      SUBROUTINE TEMPRS(PCT,TCT,PRS,TDEGC)
C*******
C******* EQUATIONS USING EITHER THE A, B OR THE C,D COEFFIECIENTS.
C******* REVISED JULY 1985 - NOW INCLUDES THE POSSIBILITY OF
C******* USED TEMP (DEGC) TO CALCULATE PRESS FROM THE CD EQN.
C******* THIS IS DONE IF BD1 = BD2
C*******
      INTEGER*4 PCT,TCT,CTREF1,CTREF2,LAB
      INTEGER*2 EQN,OVERNG,YES,TYPEQN
      REAL*4 PRS,TDEGC,TREF1,TREF2
      REAL*8 ACCNST,AC1,AC2,BDCNST,BD1,BD2,TCONST,T1,T2,T,T3
      COMMON/PCOEF/ACCNST,AC1,AC2,BDCNST,BD1,BD2,TCONST,T1,T2,T3
      COMMON/TCOEF/TREF1,TREF2,CTREF1,CTREF2,TSEC,LAB
      COMMON/PRSEQN/ EQN,OVERNG
      EQUIVALENCE (AC,A,C), (BD,B,D)
      PARAMETER (TWO16=65536, TWO24=16777216)
      PARAMETER(YES='YE',TYPEQN='CD')
C*******
C*******     CHECK FOR OVERRANGING, THEN
C*******     CALCULATE PERIOD (T) FROM PCT OF DATA SAMPLE
```

```
C*******     WHERE:  T = SAMPLING INTERVAL(IN SEC) / COUNTS
C*******
      IF(OVERNG .EQ. YES) PCT=PCT+TWO24
c*******
c******* if a zero count is found set it to unity to prevent zero divide
c******* and make a easily distinguishable spike.
c*******
if (pct.eq.0) pct=1
      T=TSEC/FLOAT(PCT)
C*******
C*******  CALCULATE TEMPERATURE:
C*******  INTERPOLATE FOR LAB TESTS:  (T-T1)/(N-N1) = (T2-T1)/(N2-N1)
C*******  IF NO LAB CALIBRATIONS, USE 'IDEAL' CONVERSION
C*******
      IF(LAB .EQ. 0) TDEGC=FLOAT(TCT)*TREF1/TSEC
      IF(LAB .EQ. 1) TDEGC=TREF1+FLOAT(TCT-CTREF1)*(TREF2-TREF1)/
     @   float((CTREF2-CTREF1))
C*******
C******* IF BD1 = BD2, EQN IS CD AND COEF CALCULATED FROM TDEGC
C******* OTHERWISE:
C*******     CALCULATE TEMP-DEPENDENT COEFFICIENTS A,B,TO
C*******     AND C,D,TO FROM TDEGC
C******* THEN DETERMINE IF EQUATION IS AB OR CD TYPE
C*******
      IF(BD1.EQ.BD2) GO TO 40
      TDEGF=TDEGC*1.80+32.
      AC=ACCNST+TDEGF*(AC1+AC2*TDEGF)
      BD=BDCNST+TDEGF*(BD1+BD2*TDEGF)
      TO=TCONST+TDEGF*(T1+T2*TDEGF)
      IF(EQN.EQ.TYPEQN) GO TO 50
C*******
C*******     CALCULATE PRESSURE FROM LINEARIZATION EQUATION:
C*******     P=A(1-TO/T) - B(1-TO/T)**2
C*******
      TOT1=1-TO/T
      PPSIA=TOT1*(A-B*TOT1)
      GO TO 100
C*******
C******* TEMP-DEPENDANT COEFS ARE CALCULATED FROM TDEGC
C******* AND EQN IS AUTOMATICALLY CD TYPE
C*******
   40 CONTINUE
      AC=ACCNST+TDEGC*(AC1+AC2*TDEGC)
      BD=BDCNST
      TO=TCONST+TDEGC*(T1+T2*TDEGC+T3*TDEGC*TDEGC)
C*******
C******* CALCULATE PRESSURE FROM PAROS EQUATION:
C******* P=C{[1-(TO/T)**2] - D[1-(TO/T)**2]**2}
C*******
   50 CONTINUE
      TOT=TO/T
      TOTSQ1=1-TOT*TOT
      PPSIA=C*(TOTSQ1 - D * TOTSQ1 * TOTSQ1)
  100 CONTINUE
```

```
C*******
C*******     CONVERT TO DBAR FROM PSIA
C*******
      PRS=PPSIA*0.68947
      RETURN
      END
C*******
C**********************************************************************
C*******
      SUBROUTINE TTMODE(LBLREC,RNMIN,RNMAX)
C******* DETERMINES MOST PROBABLE (MODAL) VALUE OF A RAYLEIGH DISTRI-
C******* BUTION P(X)=(X-XM)*EXP(-(((X-XM)**2)/2.*SD**2)
C*******    OF SAMPLE SIZE = NPTS ... BY THE METHOD OF MOMENTS, AFTER
C*******    DESPIKING RELATIVE TO QUARTILE RANGE OF SAMPLE
C******* REVISED FROM TTMOD, SEPT 1978, R. WATTS
C******* REVISED 1 NOV 1982 BY KRL:
C*******   WINDOWS THE DATA BASED ON MAX AND MIN SUPPLIED BY USER
C*******   BEFORE DETERMINING THE QUARTILE RANGE.
C******* REVISED JULY 1985, NEW COMMON BLOCK INCLUDED AND CHOPS
C******* THE DATA BEFORE PASSING IT BACK TO MAIN PROGRAM
C*******
      REAL*4 XM
      REAL*4 XX(100),UPRLIM,LOWLIM
      COMMON/COMMOD/ XX
      COMMON/MEMOCM/XMOD,NGOODP,SD,NPTS
      PARAMETER (KW=6)
      INTEGER*4 UCNT
      DATA XM/0.0/,UCNT/55/
C*******
C*******   INITIALIZE PARAMETERS.  CHECK VALUE OF NPTS.
C*******
      IF(NPTS.LT.8) GO TO 50
      SM=0.
      SD=0.0
C*******
C*******  ELIMINATE THE OUT OF RANGE POINTS BEFORE DETERMINING
C*******  THE QUARTILE RANGE OF THE RAYLEIGH DISTRIBUTION
C*******
      IBOT=1
    5 IF(XX(IBOT) .GE. RNMIN) GO TO 10
      IBOT=IBOT+1
      IF(IBOT .LE. NPTS) GO TO 5
ngoodp=0
      XMOD = RNMIN
      GO TO 45
   10 CONTINUE
      ITOP=NPTS
   15 IF(XX(ITOP) .LE. RNMAX) GO TO 20
      ITOP=ITOP-1
      IF(ITOP .GT. IBOT) GO TO 15
ngoodp=0
      XMOD = RNMAX
      GO TO 45
   20 CONTINUE
```

```
c*******
c******* bin window the data using the routine binwindow.  See documentation
c******* accomapanying the code below
c*******
call binwindow(ibot,itop)
c*******
c******* if there are less than 4 points assign xmod=the upper puns limit.
c******* This will flag ignore statistically unreliable estimates as well as
c******* out of range ones.
c*******
      NGOODP=ITOP-IBOT+1
if (ngoodp.lt.4) then
xmod=rnmax
goto 45
endif
C*******
C*******  DETERMINE THE QUARTILE RANGE - NOTE:
C*******  RAYLEIGH QUARTILE RANGE = .91 SIGMA
C*******  FOR THE TTA DETECTOR THIS SHOULD BE APPROXIMATELY 2 MSEC
C*******
      N75 = IBOT - 1 + nint(3.*float(NGOODP)/4.)
      N50 = IBOT + NGOODP/2
      N25 = IBOT - 1 + nint(float(NGOODP)/4.)
      Q = XX(N75) - XX(N25)
C*******
C*******  IF THE QUARTILE RANGE IS GREATER THAN 200. (APPROXIMATELY
C*******  10 MSEC) THEN THROW OUT THE WHOLE SAMPLE
C*******
      IF (Q .GT. 200.)  GO TO 40
C*******
C*******  THROW OUT EVERYTHING OUTSIDE THE 97TH PERCENTILE RANGE
C*******
      M=IBOT
      K=ITOP
C*******
C*******  DETERMINES UPRLIM AND LOWLIM FROM 97TH PERCENTILE RANGE
C*******  ESTIMATED FOR RAYLEIGH DISTRIBUTION, BY THE RATIO TO
C*******  QUARTILE RANGE
C*******

c******
c****** the constants in this equation were changed from
c******      UPRLIM = XX(N50) + 3.0* Q
c******      LOWLIM = XX(N50) - 1.5* Q
c****** The new constants were found using the equations:
c******  variance=(2-pi/2)sigma^2
c******  mean    = mu+sigma*(pi/2)^(.5)
c****** and an adjustment to estimate the uncertainty in the mode.
c****** This adjustment was simply a standard error of the mean
c****** error of the mean = [Variance/(N-1)]^(.5)
c****** N was taken to be 16.

      UPRLIM = XX(N50) + 1.81 * Q
      LOWLIM = XX(N50)  - 1.21 * Q
```

```
      DO 30 J=IBOT,ITOP
      IF (XX(J) .GT. UPRLIM)  K = K - 1
      IF (XX(J) .LT. LOWLIM)  M = M + 1
   30 CONTINUE
C*******
C*******  NGOODP IS THE NUMBER OF XX RETAINED FOR MODAL CALCULATION
C*******
      NGOODP=K-M+1
C*******
C*******  SM IS THE AVERAGE VALUE OF X WITHIN (M,K) KEPT
C*******
      DO 31 J=M,K
      SM=SM+XX(J)
   31 CONTINUE
      SM=SM/NGOODP
C*******
C*******  S IS THE VARIANCE (WHICH IS SCALED)
C*******
      S=0.
      DO 33 J=M,K
   33 S=S+(XX(J)-SM)**2
      S=2.*S/(NGOODP*0.86)
C*******
C*******  XM - MU, SD - SIGMA, THE RAYLEIGH WIDTH PARAMETER (SCALED VAR)
C*******
      XM=SM-SQRT(S*1.5708)
      SD=SQRT(S)
C*******
C*******  LIMIT THE RANGE OF THE DATA USING MU AND THE SD
C*******  REDEFINE UPRLIM  AND  LOWLIM
C*******
      LOWLIM= XM
      UPRLIM = XM + 4.0 * SD
      DO 34 J=M,K
      IF (XX(J) .LT. XM)  XX(J) = XM
      IF (XX(J) .GT. UPRLIM)  XX(J) = UPRLIM
   34 CONTINUE
C*******
C*******  RECALCULATE S USING THE FIXED UP DATA
C*******  SET XMOD EQUAL TO MU + SIGMA
C*******
      S=0.
      DO 35 J=M,K
   35 S=S+(XX(J)-XM)**2
      S=S/(2.*NGOODP)
      SD=SQRT(S)
      XMOD=XM+SD
      RETURN
C*******
C*******  IF ENTIRE SAMPLE IS THROWN OUT USE XMOD = XX(N25)
C*******  BASED ON EMPIRICAL EVIDENCE
C*******
   40 IF(UCNT .LE. 54) GO TO 41
      WRITE(KW,302)
```

```
  302 FORMAT('1',11X,'Q',11X,'SEQ #',11X,'XMOD',10X,'XX(25)',9X,
     @ 'XX(75)')
      WRITE(KW,310)
  310 FORMAT('+',5(5X,'__________')/)
      UCNT=0
  41  XMOD = XX(N25)
      WRITE(KW,305) Q,LBLREC,XMOD,XX(N25),XX(N75)
  305 FORMAT(1X,F15.0,I15,3F15.4)
      UCNT=UCNT+1
      RETURN
C*******
C******* ALL COUNTS OUTSIDE THE BOUNDS, WRITE MESSAGE, SET PARMS.
C*******
   45 CONTINUE
          WRITE(KW,307) ngoodp,LBLREC,XMOD
  307 FORMAT('  NGOODP =',i10,'   AT SEQ # ',I10,' XMOD = ',F20.4)
      RETURN
C*******
C*******  NOT ENOUGH POINTS TO DO ANYTHING
C*******
   50 WRITE(KW,308) NPTS
  308 FORMAT(1H ,'MISTAKE: SHOULDNT HAVE NPTS=',I4,'.LT.8')
      RETURN
      END
C*******
C**********************************************************************
C*******
      SUBROUTINE YRDAY(IYR,MNTH,IDAY,IYRDAY)
C*******
C*******  COMPUTES YEAR DAYS ( EXCEPT ON CENTURIES)
C*******
      DIMENSION ID(12)
      DATA ID(1),ID(2),ID(3),ID(4),ID(5),ID(6),ID(7),ID(8),ID(9),ID(10),
     @ID(11),ID(12)/1,32,60,91,121,152,182,213,244,274,305,335/
      IYRDAY=(ID(MNTH)-1)+IDAY
      IF((MOD(IYR,4).EQ.0).AND.(MNTH.GT.2)) IYRDAY=IYRDAY+1
      RETURN
      END
C*******
C***********************************************************************
C*******
      SUBROUTINE GMTYR (IYRDAY,IHOUR,MINUT,ISEC,GT,LOCAL,ITZON)
C*******
C*******  COMPUTES GREENWICH HOURS IN YEAR SINCE JAN 01 0000 Z
C*******  IN DECIMAL FORM - DOUBLE PRECISION
C*******
C*******  LOCAL = 0 TIME ALREADY IN GREENWICH
C*******  = 1 LOCAL TIME ... MUST CONVERT TO GREENWICH BY ADDING ITZON
C*******  ITZON IS POSITIVE FOR WEST LONGITUDE, NEGATIVE FOR EAST
C*******  E.G. 4 IS NEAR BERMUDA
C*******
      DOUBLE PRECISION GT
      IF (LOCAL) 10,10,5
    5 IHOUR = IHOUR + ITZON
```

```
   10  GT = DFLOAT((IYRDAY - 1) * 24 + IHOUR) + DFLOAT(MINUT)/60.
     @+ DFLOAT(ISEC)/3600.
       RETURN
       END
subroutine binwindow(ibot,itop)
c******* added 7/15/89
c*******
c******* This routine is a FORTRAN variation of the pascal procedure
c******* "bins" listed in Real Time Processing with Inverted Echo Sounders
c******* by Robert Petrocelli.
c*******
c******* The coding has been simplified and adapted for use within the
c******* Memode code.  A description of the variables used follows:
c*******
c******* xx     array containing the travel time counts
c******* bin    array of 96 bins representing 128 count interval from
c*******        0-8192*(1.5).  The factor of 1.5 insures that BUNSFIXed
c*******        files are suitablly processed.
c******* max    used to store the maximum number of occurrances for a bin
c******* kmax   pointer to locate max in the array bin.  Upon completion
c*******        of the routine kmax will be used to specify a range
c*******        within which the mode is most likely to be found.
c******* ibot   index of the first travel time in xx to be used.  ibot
c*******        and itop are indices found by applying the PUNS window.
c*******        Upon exit from the routine ibot and itop will contain the
c*******        new indices for the useful travel times, xx(ibot) to
c*******        xx(itop).
c******* itop   index of the last travel time to be used by this routine
c******* toplim new upper travel time limit
c******* botlim new lower travel time limit
c*******
c******* See Mr. Petrocelli's report for details.  In short this routine
c******* applies a window to refine the data before the ttmode
c******* routine. The idea is that the true surface reflections will be most
c******* probable returns and there is a time period within which all the true
c******* echo would be expected to reside.  The 13 bit range of 8192 is
c******* divided into 64 intervals. The bin containing the largest number of
c******* occurances is also most likely to be the interval within which the
c******* single travel time representative of the burst would be found.
c******* To insure that all the true echos are encompassed the adjacent
c******* intervals are included; all other bins are excluded from further
c******* processing.  The range of 3(128) counts is certainly capable of
c******* enclosing the 200 count range that the main trace on a "healthy"
c******* buns plot fits within.
c*******
integer*4 bin(100)
integer toplim,botlim
real*4 xx(100)
common/commod/xx
common/memocm/xmod,ngoodp,sd,npts

c*******
c******* don't bother if only one point
c*******
```

```
if (ibot.eq.itop) return
c*******
c******* initialize max and array bin
c*******
max=0
do 40 i=1,100
bin(i)=0
40 continue
c*******
c*******  loop thru and increment the number occurances within a particular
c*******  counts interval. Example: If ixx(i)=394 then k=4 and bin(4) is
c*******  increased by one.  If ixx(i)=128 then k=1 and bin(1)=bin(1)+1.
c*******  Note that the first term of k is integer division.
c*******
do 10 i=ibot,itop
k=(int(xx(i))-1)/128 +1
bin(k)=bin(k)+1
c*******
c*******  store the current maximum number of occurrances and its location
c*******
if (bin(k).gt.max) then
max=bin(k)
kmax=k
endif
10 continue
c******
c****** set the travel time limits
c******
toplim=(kmax+1)*128
botlim=(kmax-2)*128
c******
c****** use these limits to find indices of fist and last points in the
c****** new window.
c******
c****** clamp down to find the bottom of the window.
c******
        do 20 i=ibot,itop
if (xx(i).ge.botlim) then
ibot=i
goto 100
endif
20 continue
c******
c****** do the similar operation to find the top of the window.
c******
100 do 30 i=itop,ibot,-1
if (xx(i).le.toplim) then
itop=i
return
endif
30 continue
```

```
end
```

## 3.5 DETIDE_AUG90.FOR

```
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c%%%%%%%  detide_aug90.for
c%%%%%%%   -
c%%%%%%% revised from detide_jul88.for to deal with *.fill files from
c%%%%%%% fill_aug90.for.  These differ from detide_jul88.for in that they
c%%%%%%% contain  only travel time and yearhour ( no tmp, prs, or am col-
c%%%%%%% umns).
c%%%%%%%
c%%%%%%% Fields 4-aug-90
C*******
C*******                    DETIDE.JULY88.FOR
C*******
C******* PROGRAM TO DETIDE THE MEDIAN/MODE VALUES OF TRAVEL TIMES.
C******* DETERMINES THE TIDAL HEIGHT IN CM AND CONVERTS IT TO SECONDS.
C******* IT IS ASSUMED THAT THE DATASET HAS BEEN PROCESSED THROUGH THE
C******* FILL PROGRAM, SO THAT THERE ARE NO GAPS.  THE OUTPUT DATASET
C******* CONTAINS, THE MEASURED TAU'S, THE DETIDED TAU'S AND THE TIDE,
C******* AS WELL AS THE OTHER INPUT PARAMETERS.
C*******
C******* FORTRAN UNIT NUMBERS DESIGNATED AS FOLLOWS:
C*******     KR  (UNIT 15) CONTROL CARD INPUT FILE
C*******     KR1 (UNIT 18) INPUT DATASET FROM FILL
C*******     KW  (UNIT 11) OUTPUT USERS LOG
C*******     KW1 (UNIT 17) OUTPUT DATA FILE
C*******
      INTEGER*4 YEAR,TOTREC,HUNDRD,RECOUT
      CHARACTER*60 HEADR
      CHARACTER*10 LOCN
      REAL*4 TIME(200),DELT
      REAL*4 TTB(200),ZO,MTIDCM,TTID(200)
      REAL*4 XODTID(200)
      REAL*8 DTIME
C*******
      COMMON/TIDCON/ H(8), PHI(8),VU(8),FNODE(8)
      COMMON/TPARMS/ZO,IYR,ICALVU
C*******
      PARAMETER (HUNDRD=200)
      PARAMETER (KR=15,KW=11,KW1=17,KR1=18)
      NAMELIST/CARD1/HEADR
      NAMELIST/CARD2/NFIRST,NLAST,IYR,DELT
      NAMELIST/CARD3/H
      NAMELIST/CARD4/PHI
      NAMELIST/CARD5/FNODE
      NAMELIST/CARD6/VU
      NAMELIST/CARD7/CBAR,PFACTR,LOCN
      DATA ICALVU/0/
      DATA CBAR/1510./,PFACTR/1.0/,LOCN/'N.U.'/
      DATA MTIDCM/0.0/,TTID/200*0.0/
      DATA ZO/0.0/
      DATA H/8*0.0/,PHI/8*0.0/,FNODE/8*1.0/,VU/8*0.0/
      DATA RECOUT/0/,IPRNT/0/
C*****
```

```
C***** OPEN I/O UNITS AND FILES
C*****
       OPEN(UNIT=KR,STATUS='OLD',FORM='FORMATTED',READONLY)
       OPEN(UNIT=KR1,STATUS='OLD',FORM='FORMATTED',READONLY)
       OPEN(UNIT=KW,STATUS='NEW',FORM='FORMATTED')
       OPEN(UNIT=KW1,STATUS='NEW',FORM='FORMATTED')
C******
C******   READ CONTROL PARAMETERS
C******   NFIRST:  THE REC # OF THE FIRST 'GOOD' RECORD
C******   NLAST:  LAST REC # TO BE PROCESSED
C******
      READ(KR,NML=CARD1)
      READ(KR,NML=CARD2)
      READ(KR,NML=CARD3)
      READ(KR,NML=CARD4)
      READ(KR,NML=CARD5)
      READ(KR,NML=CARD6)
      READ(KR,NML=CARD7)
C*****
      WRITE(*,42)
 42   FORMAT(1X,//,5X,' DETIDE IS NOW PROCESSING YOUR DATA!!'//)
C*******
C*******   PRINT CONTROL PARAMETERS
C*******
      WRITE(KW,400)
  400 FORMAT(1H1,T50,'* * *  PROGRAM  DETIDE  * * *',//)
      WRITE(KW,405) HEADR
  405 FORMAT(/T35,A60)
      WRITE(KW,410) NFIRST,NLAST,CBAR,PFACTR,LOCN
  410 FORMAT(//T15,'NFIRST',T26,'NLAST',T37,
     @'CBAR',T45,'PFACTR',T57,'LOCN',
     @/,'+',T11,5(2X,'________'),
     @//,T10,2I10,2F10.2,6X,A4)
      WRITE(KW,415) IYR,DELT
  415 FORMAT(//8X,'IYR =',I10//8X,'DELT=',F12.7)
      WRITE(KW,420)H,PHI,VU,FNODE
  420 FORMAT(//T10,'TIDAL PARAMETERS ',
     @//T17,'M2',T27,'N2',T37,'S2',T47,'K2',T57,
     @'K1',T67,'O1',T77,'P1',T87,'Q1',
     @//T9,'H',8F10.2,
     @/T7,'PHI',8F10.2,
     @/T6,'V0+U',8F10.3,
     @/T9,'F',8F10.3//)
      WRITE(KW,425)
  425 FORMAT(//T40,'FIRST RECORD OF EACH BLOCK PROCESSED',
     @//T16,'TTB',T26,'XODTID',T37,'TTID',T48,'TIME',
     @/,'+',T12,2X,'________',3(3x,'________'),//)
C*******
C******* COMPUTE TRAVEL TIME CONVERSION FACTOR, TTCONV CONVERTS TIDE
C******* FROM CM TO SEC.  CBAR IN (M/SEC) IS AVERAGE SOUND VELOCITY
C******* FROM MATTHEWS TABLES FOR THE LOCATION AND DEPTH OF THE IES.
C******* PFACTR  = 1 + DEPTH * 1.1E-5,   WHERE 1.1E-5=(1/C)(DC/DP)
C******* DEFAULT FOR PFACTR IS 1.0
C*******
```

```
      TTCONV = 2.0E-2 / (CBAR * PFACTR)
C*******
C******* READ DATA IN BLOCKS OF 200 TO IMPROVE I/O EFFICIENCY.
C*******
C*******
C******* COMPUTE TOTAL # OF RECORDS TO BE READ, AND THE
C******* CORRESPONDING NUMBER OF BLOCKS, PLUS A REMAINDER.
C*******
   25 CONTINUE
      TOTREC = NLAST-NFIRST + 1
      NBLK = TOTREC/HUNDRD
      NGET = MOD( TOTREC , HUNDRD )
      IF( NBLK .LT. 1) NGET = TOTREC
      NGO = HUNDRD
      NRDBLK = 0
C*******
C******* READ BLOCK OF DATA, CHECKING THAT THERE ARE AT LEAST 200
C*******
   30 IF(NRDBLK .GE. NBLK) NGO = NGET
      IF(NGO .EQ. 0) GO TO 55
      READ(KR1,430,END=999) (TTB(I),TIME(I),I=1,NGO)
  430 FORMAT(2E15.7)
      NRDBLK = NRDBLK + 1
C*******
C*******  PROCESS ONE BLOCK OF DATA AT A TIME
C*******
      DO 45 J=1,NGO
C*******
C*******   CONVERT TIME TO DOUBLE PRECISION
C*******
   40 CONTINUE
      DTIME = DBLE(TIME(J))
C*******
C******* NOW COMPUTE TIDAL HEIGHT
C*******
      CALL TIDE(DTIME,MTIDCM)
C*******
C******* ALL TRAVEL TIMES ARE IN SECONDS, SO SCALE TIDE TO SECONDS AND
C******* SUBTRACT FROM THE INPUT DATA
C*******
      TTID(J) = (TTCONV * MTIDCM)
      XODTID(J) = TTB(J) - TTID(J)
   45 CONTINUE
C*******
C*******  PRINT FIRST RECORD OF EACH BLOCK TO OUTPUT LOG
C*******
      WRITE (KW,435) TTB(1),XODTID(1),TTID(1),TIME(1)
  435 FORMAT(/T10,F11.5,1X,2F11.5,F10.2)
      IPRNT = IPRNT + 1
C*******
C*******   DUMP IT ALL ON THE DISK OUTPUT
C*******
      WRITE(KW1,440) (TTB(J),XODTID(J),TTID(J),TIME(J),J=1,NGO)
  440 FORMAT(4E15.7)
```

```
      RECOUT = RECOUT + NGO
   50 IF(NGET .LT. NGO) GO TO 30
C*******
C*******   END OF PROCESSING - WRAP UP
C*******
   55 CONTINUE
      WRITE(KW,450) RECOUT
  450 FORMAT(//T10,' *** PROCESSING ENDED AT SPECIFIED RECORD--',
     @I6,'  RECORDS WRITTEN ON UNIT 12'//)
      WRITE(*,450) RECOUT
      WRITE(*,43)
      STOP
 999  WRITE(KW,455) RECOUT,I
 455  FORMAT(//T10,'>>>> ERROR: UNEXPECTED END OF DATA <<<<',
     @/T10,I6,'  RECORDS PROCESSED',I6,' ADDITIONAL RECORDS',
     @' READ BEFORE EOF'//)
C*****
      WRITE(*,455) RECOUT,I
      WRITE(*,43)
 43   FORMAT(6X,' FINISHED!!'//)
      STOP
      END
C*******
C**********************************************************************
C*******
      SUBROUTINE TIDE(GYRHR,ZT)
C*******
C*******   TIDAL COMPONENTS ORDERED: M2, N2, S2, K2, K1, O1, P1, Q1
C*******   REQUIRES HALF AMPLITUDES: H(L), CM
C*******   PHASES:  PHI(L) DEGREES, GREENWICH EPOCH
C*******   GYRHR = GREENWICH HOURS SINCE THE BEGINNING OF IYEAR
C*******   VOANDU IS THE ARGUMENT OF EQUILIBRIUM TIDE (GEOPOTENTIAL)
C*******   AT THE STARTING TIME AT GREENWICH MERIDIAN (VO + U
C*******   IN TABLE 15 OF COAST AND GEODETIC SURVEY SPECIAL PUB. 98).
C*******
C*******
C*******   METHOD:  ZT = ZO + SUM (H(L) * COS(ARG(L))
C******* WHERE:
C******* ARG(L) = TPI * (FREQ(L) * TIME + (VU(L) - PHI(L)) / 360.0)
C*******
C******* VU(L) IS THE ARGUMENT (DEGREES) OF EQUILIBRIUM TIDAL
C******* CONSTITUENT L AT REFERENCE TIME = NDAYS
C*******
C******* ZT IS THE CALCULATED TIDE
C*******
C******* NOTE:  GIVES CAREFUL ATTENTION TO USING DOUBLE PRECISION
C******* ONLY WHERE NECESSARY
C*******
      COMMON/TIDCON/ H(8),PHI(8),VU(8),FNODE(8)
      COMMON/TPARMS/ZO,IYEAR,ICALVU
      REAL*4 VOANDU(8),PCYCLS(8)
      INTEGER*4 CALLCT
      DOUBLE PRECISION FREQ(8),GYRHR,DAYS,FLOAT,DCYCLS
      PARAMETER (TPI=6.2831853,KW=1)
```

```
      DATA CALLCT/0/
      DATA FREQ(1)/1.9322736D0/,
     @     FREQ(2)/1.8959820D0/,
     @     FREQ(3)/2.0D0/,
     @     FREQ(4)/2.0054758D0/,
     @     FREQ(5)/1.0027379D0/,
     @     FREQ(6)/0.9295357D0/,
     @     FREQ(7)/0.9972621D0/,
     @     FREQ(8)/0.8932441D0/
      DATA VOANDU/8.3,65.7,0.0,217.5,18.9,345.2,349.8,42.6/
C*******
C*******  COUNT NUMBER TIMES THIS SUBROUTINE HAS BEEN CALLED.
C*******  CHECK TO SEE IF VU MUST BE CALUCLATED (ICALVU=1)
C*******  OR IF IT WAS SUPPLIED (ICALVU=0).
C*******
      CALLCT=CALLCT+1
      IF (ICALVU .EQ. 0) GO TO 25
C*******
C******* VU MUST BE CALCULATED:
C*******   CALCULATE # OF DAYS FROM 1 JAN 1900 TO 1 JAN IYEAR
C*******   LEAP DAYS ON YEARS DIVISIBLE BY 4 EXCEPT 1900
C*******   (CENTURY NOT LEAP YR UNLESS DIVISIBLE BY 400).
C*******   CALCULATE # OF CYCLES PER DAY AT EACH FREQUENCY,
C*******   USE ONLY FRACTIONAL NUMBER OF CYCLES.
C*******
   10 LPDYS = (IYEAR - 1900) / 4
      NDAYS = (IYEAR - 1900) * 365 + LPDYS
      DO 15 L = 1, 8
      DCYCLS = FREQ(L) * NDAYS
      DCYCLS = DCYCLS - FLOAT(IDINT(DCYCLS))
      VU(L) = SNGL(DCYCLS) * 360.0 + VOANDU(L)
      IF (VU(L) .GE. 360.) VU(L) = VU(L) - 360.
   15 CONTINUE
      ICALVU = 0
      WRITE(KW,400) IYEAR,(VU(L),L=1,8)
  400 FORMAT(//1H0,'EQUILIBRIUM TIDE ARGUMENTS (DEGREES) ',
     @' AT THE BEGINNING OF ',I4,8F8.1//1H0)
C*******
C******* VU ALREADY CALCULATED:
C*******  IF FIRST CALL TO SUBROUTINE, CALCULATE THE DIFFERENCES
C*******  BETWEEN THE PHASES VU AND PHI.  THEN CONVERT TO CYCLES/DAY.
C*******
   25 CONTINUE
      IF (CALLCT .GT. 1) GO TO 35
      DO 30 L=1,8
      PCYCLS(L) = (VU(L) - PHI(L)) / 360.0
   30 CONTINUE
C*******
C******* MAIN LOOP FOR CALCULATING THE TIDE AT TIME=GYRHR
C*******
   35 CONTINUE
      DAYS = GYRHR / 24.0D0
      ZT = Z0
      DO 40 L=1, 8
```

```
      DCYCLS = FREQ(L) * DAYS
      DCYCLS = DCYCLS - FLOAT(IDINT(DCYCLS))
      ARGL = (SNGL(DCYCLS) + PCYCLS(L)) * TPI
      ZT = ZT + FNODE(L) * H(L) * COS(ARGL)
   40 CONTINUE
      RETURN
      END
```

## 3.6 DESPIKE_AUG90.FOR

```
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c%%%%%%%   despike_aug90.for
c%%%%%%%   -
c%%%%%%%   differs from despike_jul88.for in logical unit numbers
c%%%%%%%   and the number of columns in the input file.  This version
c%%%%%%%   is applied to a four column travel time input file.
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C*******
C*******                   DESPIKE.JULY88.FOR
C*******
C*******   PROGRAM TO DESPIKE THE DETIDED TRAVEL TIMES.   SPIKES ARE
C******* IDENTIFIED IN TWO WAYS:  1) THE DATA MUST BE WITHIN A SPECIFIED
C******* RANGE OF GOOD TRAVEL TIMES; AND 2) THE CHANGE BETWEEN TWO
C******* ADJACENT MEASUREMENTS MUST BE LESS THAN A SPECIFIED RATE.  IF
C******* EITHER OF THESE TWO CRITRIA IS NOT MET, THE DATA POINT IS
C******* REPLACED WITH AN INTERPOLATED VALUE.
C*******
C*******   PROGRAM ORIGINALLY WRITTEN BY BY J. GUNN 1978,
C******* BUT HAS BEEN REWRITTEN SEVERAL TIMES SINCE 1981 BY K. TRACEY.
C******* CONVERTED TO VAX BY SLW
C*******
C*******   NOTE, THE SAV... ARRAYS MUST BE DIMENSIONED THE SAME AS OR
C******* GREATER THAN MAXBAD.
C*******
C******* I/O UNITS:
C*******     KR  (UNIT 17) CONTROL CARD FILE
C*******     KR1 (UNIT 19) INPUT DATASET FROM DETIDE
C*******     KW  (UNIT 11) OUTPUT USERS LOG
C*******     KW1 (UNIT 18) OUTPUT DATASET
C*******     KW2 (UNIT 16) OUTPUT LISTINGS FILE
C*******
      INTEGER*4 IRECCT, BADREC, IBADCT, NBAD
      INTEGER*4 FLAGCT
      LOGICAL FLAG
      CHARACTER*80 HEADR
      INTEGER*4 MAXBAD
      REAL*4 KNT1
      REAL*4 SAVXO(100), SAVXOD(100),INBTWN
      REAL*4 XO,TIME,TIDE
      REAL*4 XOFIL, XODFIL
      REAL*4 SAVTID(100),SAVTIM(100),KNT1S
      REAL*8 X1SAV(100),XODIF,XOMAX
      REAL*8 RNAVG1,  SMAX1
      PARAMETER (KR1=19, KW=11, KW1=18, KR=17, KW2=16)
      PARAMETER (ON='<===',OFF='    ')
      NAMELIST/CARD1/HEADR
      NAMELIST/CARD2/TINTVL,VMAX,VMIN
      NAMELIST/CARD3/SLOPE1,RNAVG1,LAVG1
      DATA IRECCT/0/, LINECT/55/
      DATA IBADCT/0/, NBAD/0/, FLAGCT/0/, MAXBAD/100/
C*****
C***** OPEN I/O UNITS AND FILES
```

```
C*****
        OPEN(UNIT=KR,STATUS='OLD',FORM='FORMATTED',READONLY)
        OPEN(UNIT=KR1,STATUS='OLD',FORM='FORMATTED',READONLY)
        OPEN(UNIT=KW,STATUS='NEW',FORM='FORMATTED')
        OPEN(UNIT=KW1,STATUS='NEW',FORM='FORMATTED')
        OPEN(UNIT=KW2,STATUS='NEW',FORM='FORMATTED')
C*****
C*****  READ AND WRITE IN THE CONTROL CARDS
C*****
      READ(KR,NML=CARD1)
      READ(KR,NML=CARD2)
      READ(KR,NML=CARD3)
      write(*,42)
 42   format(1x,//,5x,'DESPIKING your data set---> Pondering ',/)
      WRITE(KW,608) HEADR
 608  FORMAT(//45X,'.............. DESPIKE ..............',
     @/'0',A80)
      WRITE(KW,610) TINTVL, VMAX, VMIN,
     @SLOPE1, RNAVG1, LAVG1
  610 FORMAT(//5X,'CONTROL CARDS',
     @T25,9X,'TINTVL',11X,'VMAX',11X,'VMIN'/
     @T25,3(5X,F10.4)//
     @T25,9X,'SLOPE1',9X,'RNAVG1',10X,'LAVG1'/
     @T25,2(5X,F10.4),5X,I10//)
C*******
C*******  INITIALIZE VARIABLES
C******* KNT1 AND KNT1S ARE INDEXES OF THE X1SAV ARRAY
C******* LAST1 IS PREVIOUS VALUE OF KNT1
C******* RLAVG1 IS NUMBER OF POINTS IN RUNNING AVERAGE
C******* SMAX1 IS MAXIMUM ALLOWED CHANGE IN SECONDS PER SAMPLING PERIOD
C*******
      KNT1 = (LAVG1 + 1.0)/2.0
      KNT1S = KNT1
      LAST1 = LAVG1
      RLAVG1 = LAVG1
      SMAX1 = SLOPE1*TINTVL
C*******
C*******  IF THE INITIAL RUNNING AVERAGES WERE NOT SUPPLIED,
C*******  COMPUTE THEM FROM THE FIRST LAVG1 POINTS IN THE DATA SET
C*******  NOTE THAT THESE POINTS WILL NOT BE DESPIKED.
C*******
      IF (RNAVG1 .NE. 0.0) GO TO 30
      DO 20 I=1,LAVG1
      READ(KR1,400,END=900)X0,XODTID,TIDE,TIME
      IRECCT = IRECCT + 1
      WRITE(KW1,400) X0,XODTID,TIDE,TIME
      X1SAV(I) = XODTID / RLAVG1
      RNAVG1 = RNAVG1 + X1SAV(I)
   20 CONTINUE
      GO TO 100
C*******
C*******  IF INITIAL RUNNING AVERAGES WERE SUPPLIED,
C*******  INITIALIZE THE PREVIOUS POINTS ARRAY TO THE RUNNING AVERAGES
C*******
```

```
   30 CONTINUE
      DO 40 I=1,LAVG1
      X1SAV(I) = RNAVG1 / RLAVG1
   40 CONTINUE
C*******
C*******         MAIN PROCESSING LOOP
C*******   REMOVE THE SPIKES FROM THE DATA
C*******
  100 CONTINUE
      READ (KR1,400,END=900) XO,XODTID,TIDE,TIME
  400 FORMAT(4E15.7)
      IRECCT = IRECCT + 1
C*******
C******* TEST EACH DETIDED POINT
C******* SEE IF IT IS OUTSIDE THE LIMITS BEFORE CHECKING THE SLOPE
C*******
      IF (XODTID .GE. VMAX)  GO TO 105
      IF (XODTID .LE. VMIN)  GO TO 105
      XODIF = XODTID - RNAVG1
      XOMAX = KNT1S * SMAX1
      IF (DABS(XODIF) .GT. XOMAX) GO TO 105
      GO TO 110
C*******
C******* FOR BAD POINTS, OPEN UP THE WINDOW AND SAVE THE VALUES.
C******* TO BE WRITTEN OUT LATED WITH THE INTERPOLATED VALUES.
C*******
  105 CONTINUE
      KNT1S = KNT1S + 1.0
      NBAD = NBAD + 1
      IF (NBAD .EQ. 1)  BADREC = IRECCT
      IF (NBAD .GT. MAXBAD)  GO TO 2000
      SAVXO (NBAD) = XO
      SAVXOD(NBAD) = XODTID
      SAVTID(NBAD) = TIDE
      SAVTIM(NBAD) = TIME
      GO TO 100
C*******
C******* IF THE POINT WAS O.K., THEN
C******* RESET THE WINDOW, ACCOUNTING FOR LENGTHENING IF FILLED
C*******
  110 CONTINUE
      KNT1S = KNT1 + NBAD/2.0
C*******
C******* IF PREVIOUS POINT(S) WERE GOOD - SKIP THIS LOOP
C******* IF NECESSARY, WRITE A NEW HEADER FOR THE LOG PAGE
C*******
      IF (NBAD .EQ. 0)  GO TO 115
      IF(LINECT .LE. 54) GO TO 77
      WRITE(KW2,72)
   72 FORMAT('1',7X,'RNAVG',8X,'NBAD',8X,'REC#',8X,'XOLD',7X,
     @'XOFIL',4X,'XDTIDOLD',4X,'XDTIDFIL',8X,'TTID',10X,'TIME',
     @/'+',9('  __________'),'__')
      LINECT = 0
   77 WRITE(KW2,632) RNAVG1, NBAD
```

```
  632 FORMAT(/3X,F10.6,2X,I10)
      PRVXOD = X1SAV(LAST1) * RLAVG1
      DLTXOD = XODTID - PRVXOD
      XODINC = DLTXOD / (NBAD + 1)
C*******
C*******          INTERPOLATING LOOP
C******* FILL IN THE BAD POINTS WITH INTERPOLATED POINTS
C*******
      DO 114 I = 1, NBAD
C*******
C******* IF VALUE OF BAD POINT IS BETWEEN TWO GOOD ONES - SET FLAG
C*******
      FLAG = OFF
      INBTWN = (XODTID - SAVXOD(I)) * (SAVXOD(I) - PRVXOD)
      IF(INBTWN .LE. 0.0) GO TO 112
  113 FLAG = ON
      FLAGCT = FLAGCT + 1
C*******
C******* .....UPDATE THE FIFO RUNAVG STACK WITH FLAGGED POINTS ......
C*******
      LAST1 = LAST1 + 1
      IF (LAST1 .GT. LAVG1) LAST1 = 1
      RNAVG1 = RNAVG1 - X1SAV(LAST1)
      XNOW = DBLE(SAVXOD(I)) / RLAVG1
      RNAVG1 = RNAVG1 + XNOW
      X1SAV(LAST1) = XNOW
C*******
C*******  ....END OF UPDATING FIFO STACK WITH INTERPOLATED DATA....
C*******
  112 CONTINUE
      XODFIL = PRVXOD + XODINC * I
      XOFIL = XODFIL + SAVTID(I)
C*******
C******* WRITE INTERPOLATED POINTS TO OUTPUT LOG AND DATASET
C*******
      IF(I .GT. 1) GO TO 78
      WRITE(KW2,634) BADREC, SAVXO(I), XOFIL,
     @SAVXOD(I), XODFIL, SAVTID(I), SAVTIM(I),FLAG
  634 FORMAT('+',26X,I10,5(2X,F10.6),2X,F12.5,2X,A4)
      LINECT = LINECT + 2
      GO TO 79
   78 WRITE(KW2,635) BADREC, SAVXO(I), XOFIL,
     @SAVXOD(I), XODFIL, SAVTID(I), SAVTIM(I),FLAG
  635 FORMAT(' ',26X,I10,5(2X,F10.6),2X,F12.5,2X,A4)
      LINECT = LINECT + 1
   79 WRITE(KW1,400) XOFIL, XODFIL, SAVTID(I), SAVTIM(I)
      BADREC = BADREC + 1
  114 CONTINUE
C*******
C******* INCREMENT THE COUNTERS
C*******
      IBADCT = IBADCT + NBAD
      NBAD = 0
  115 CONTINUE
```

```
C*******
C******* SAVE THE LAST LNGAVG POINTS IN A FIFO STACK
C******* AND UPDATE THE RUNNING AVERAGES
C******* NOTE:  THE STACK CONTAINS THE VALUE OF EACH POINT DIVIDED BY
C******* THE NUMBER OF POINTS IN THE RUNNING AVERAGE
C******* THIS WAY WE ONLY HAVE TO DIVIDE ONCE PER POINT
C*******
      LAST1 = LAST1 + 1
      IF (LAST1 .GT. LAVG1) LAST1 = 1
      RNAVG1 = RNAVG1 - X1SAV(LAST1)
      X1SAV(LAST1) = DBLE(XODTID) / RLAVG1
      RNAVG1 = RNAVG1 + X1SAV(LAST1)
C*******
C******* WRITE THE GOOD POINTS TO THE OUTPUT DATA SET
C******* THEN CONTINUE TH PROCESSING LOOP
C*******
      WRITE(KW1,400) XO,XODTID,TIDE,TIME
      GO TO 100
C*******
C******* IF TOO MANY CONSECUTIVE BAD RECORDS -
C******* PRINT MESSAGE AND STOP
C*******
 2000 CONTINUE
      WRITE(KW,680) MAXBAD
  680 FORMAT(//5X,'MORE THAN',I4,' CONSECUTIVE BAD POINTS FOUND',
     @//5X,'*****  RUN TERMINATED  *****')
      WRITE(*,680) MAXBAD
      STOP 999
C*******
C******* WRAP UP  -  WRITE INFO TO OUTPUT LOG
C*******
  900 CONTINUE
      WRITE(KW,690) IRECCT, IBADCT, FLAGCT
  690 FORMAT(//5X,I5,' RECORDS WERE PROCESSED',
     @//5X,I5,' BAD POINTS WERE REPLACED',
     @//5X,I5,' REPLACEMENTS WERE FLAGGED')
      write(*,43)
 43   format(1x,//,5x,'Pondering Finished!')
      STOP
      END
```

### 3.7 SEACOR_AUG90.FOR

```
c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c%%%%%%% seacor_aug90.for
c%%%%%%%
c%%%%%%% revised in accordance with the aug90 changes in the
c%%%%%%% processing tree.  Seacor_aug90 is applied to a file
c%%%%%%% without temp, press, and amb columns
c%%%%%%%
c""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
c"""""""""  23-Nov-1990 revised to include the new seasonal
c"""""""""  corrections for my Stephen, Karen, and DRW.  The
c"""""""""  correction factors used are location specific.
c"""""""""  There are three regions.  The SYNOP data falls into two
c"""""""""  of the regions; The inlet array is region 1, and the
c"""""""""  central array is in region 2.
c"""""""""
c"""""""""  The control file now requires a third namelist containing
c"""""""""  the numerical specification of the region (1 or 2).
c""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
C*******
C*******                      SEACOR.JULY88.FOR
C*******
C*******   A SEASONAL CORRECTION FACTOR IS ADDED TO THE MEASURED AND
C*******   DETIDED TRAVEL TIMES.  THE CORRCTION FACTOR FOR EACH
C*******   SAMPLING PERIOD IS CALCULATED BY A LINEAR INTERPOLATION
C*******   BETWEEN THE ARRAY OF MONTHLY CORRECTION FACTORS (CF).
C*******
C*******   FOR THIS VERSION, THE VALUES IN ARRAY CF ARE FOR THE GULF
C*******   STREAM REGION.  THEY WERE CALCULATED USING ISELIN'S (1940)
C*******   HYDROGRAPHIC DATA.
C*******
C*******   I/O UNITS:
C*******     KRCTRL (UNIT 1) - CONTROL FILE
C*******     KWLOG  (UNIT 2) - OUTPUT USER'S LOG
C*******     KRIES  (UNIT 3) - IES INPUT DATA FILE
C*******     KWIES  (UNIT 4) - OUTPUT FILE OF SEASONALLY CORRECTED
C*******                       IES DATA
C*******
      CHARACTER*2 YES/'YE'/,FRSTYR/'NO'/,SCNDYR/'NO'/
      CHARACTER*80 HEADR
      INTEGER*4 HI, LOW
integer region
      REAL*4 TT(200),TTDTID(200),TID(200),TIME(200)
c      REAL*4 TMP(200),PRS(200),AMB(200)
      REAL*4 TTSCF(200),TDTSCF(200)
      REAL*4 MNTH(24),CF(24),CF1(24),CF2(24),CF3(24),DSLOPE(24)
      REAL*4 CFDIF,MDIF,YREND(2),YRHR
      REAL*8 DEL,LSTDEL,TDIF,SCF
      REAL*8 DYRHR, DHMNTH, DLMNTH
      PARAMETER(KRCTRL=1,KWLOG=2,KRIES=3,KWIES=4)
      NAMELIST/CARD1/HEADR
      NAMELIST/CARD2/NPTS,NOYRS,FRSTYR,SCNDYR
      namelist/card3/region
```

```
      DATA NTOTL/0/,YREND/8760.0,8760.0/
C*******
C*******  INITIALIZE THE YEARHOUR AND CORRECTION FACTOR ARRAYS
C*******  PROGRAM ASSUMES THAT THE YEARHOURS ARE FOR THE FIRST
C*******  DAY OF EACH MONTH.
C*******  THESE VALUES ARE FOR THE GULF STREAM REGION.
C*******
      DATA MNTH/0., 744., 1416., 2160., 2880., 3624.,
     @          4344., 5088., 5832., 6552., 7296., 8016.,
     @          0., 744., 1416., 2160., 2880., 3624.,
     @          4344., 5088., 5832., 6552., 7296., 8016./
      DATA CF1/0.000551553, 0.000143760, 0.0, 0.000244602,
     /        0.000581494, 0.000730828, 0.000743656, 0.000727823,
     /        0.000746362, 0.000850607, 0.000960953,0.000886401,
     /        0.000551553, 0.000143760, 0.0, 0.000244602,
     /        0.000581494, 0.000730828, 0.000743656, 0.000727823,
     /        0.000746362, 0.000850607, 0.000960953,0.000886401/
      DATA CF2/0.00069029, 0.00028889, 0.0, 0.00001659,
     /        0.00034654, 0.00080489, 0.00115238, 0.00134841,
     /        0.00143492, 0.00135541, 0.00116881, 0.00097876,
     /        0.00069029, 0.00028889, 0.0, 0.00001659,
     /        0.00034654, 0.00080489, 0.00115238, 0.00134841,
     /        0.00143492, 0.00135541, 0.00116881, 0.00097876/
      DATA CF3/0.00038513, 0.00012636, 0.0, 0.00006714,
     /        0.00046289, 0.00107709, 0.00155502, 0.00174333,
     /        0.00178230, 0.00169593, 0.00135379, 0.00082930,
     /        0.00038513, 0.00012636, 0.0, 0.00006714,
     /        0.00046289, 0.00107709, 0.00155502, 0.00174333,
     /        0.00178230, 0.00169593, 0.00135379, 0.00082930/
C*******
C*******  OPEN THE INPUT AND OUTPUT DATASETS
C*******
      OPEN(UNIT=KRIES,STATUS='OLD',FORM='FORMATTED',READONLY)
      OPEN(UNIT=KWIES,STATUS='NEW',FORM='FORMATTED')
C*******
C*******   READ CONTROL CARD FILE
C*******
      READ(KRCTRL,NML=CARD1)
      READ(KRCTRL,NML=CARD2)
      READ(KRCTRL,NML=CARD3)
if (region.eq.1) then
do 67 i=1,24
cf(i)=cf1(i)
67 continue
else if (region.eq.3) then
do 68 i=1,24
cf(i)=cf3(i)
68 continue
else
do 69 i=1,24
cf(i)=cf1(i)
69 continue
endif
```

```
      WRITE(*,42)
 42   FORMAT(1X,//,5X,' SEACOR HAS TAKEN OVER AND IS PROCESSING.',/)
C*******
C*******   IF A LEAP YEAR, ADJUST THE BEGINNING TIMES OF EACH MONTH
C*******
      IF (FRSTYR .EQ. YES) THEN
        YREND(1)=8784.0
        DO 100 LP=3,12
        MNTH(LP)=MNTH(LP)+24.0
  100   CONTINUE
      ELSE IF (SCNDYR .EQ. YES) THEN
        YREND(2)=8784.0
        DO 110 LP=15,24
        MNTH(LP)=MNTH(LP)+24.0
  110   CONTINUE
      END IF
C*******
C*******  IF DATASET SPANS TWO YEARS, THE FIRST YEAR WILL HAVE
C*******  NEGATIVE YEARHOURS.  SO RESET THE BEGINNING MONTHS
C*******
      IF (NOYRS .EQ. 2) THEN
        DO 115 LP=1,12
        MNTH(LP)=MNTH(LP)-YREND(1)
  115   CONTINUE
      END IF
C*******
C*******   WRITE HEADER INFO TO USER'S LOG
C*******
   20 WRITE(KWLOG,405) HEADR
  405 FORMAT('1',T35,'***** SEASONAL CORRECTION FACTOR PROGRAM ',
     @      '*****'//'0',A80)
      WRITE(KWLOG,410) NPTS,NOYRS, FRSTYR, SCNDYR
  410 FORMAT('0',I10,' SAMPLING PERIODS ARE TO BE READ',
     @ /'0 THE DATASET SPANS',I2,' YEAR(S)',
     @ /'  IS THE FIRST YEAR A LEAP YEAR? ',A4,
     @ /'  IS THE SECOND YEAR A LEAP YEAR?',A4)
      WRITE(KWLOG,415)
  415 FORMAT(///6X,'TABLE OF SEASONAL CORRECTION FACTORS',
     @     /'0',4('  YRHR  ',5X,'   CF   ',5X),
     @     /'+',4('________',5X,'________',5X))
 10   WRITE(KWLOG,420) (MNTH(I),CF(I),MNTH(I+6),CF(I+6),
     @MNTH(I+12),CF(I+12),MNTH(I+18),CF(I+18),I=1,6)
 11   CONTINUE
  420 FORMAT(4(F8.0,5X,F8.4,5X))
C*******
C*******   CALCULATE THE SLOPES TO BE USED DURING INTERPLOATION
C*******
      DO 40 I=1,24
      IF(I-24 .NE. 0) THEN
        CFDIF = CF(I+1) - CF(I)
        MDIF= MNTH(I+1) - MNTH(I)
        DSLOPE(I)=CFDIF/MDIF
      ELSE
        CFDIF=CF(1)-CF(I)
```

```
          MDIF= YREND(NOYRS) - MNTH(I)
          DSLOPE(I)=CFDIF/MDIF
        END IF
   40 CONTINUE
C*******
C*******  INTIALIZE THE PARAMETERS TO BE USED FOR READING THE DATA
C*******
        IF (NPTS .LT. 200) THEN
           NGO = NPTS
           NLEFT = 0
        ELSE
           NGO = 200
           NLEFT = NPTS
        END IF
C*******
C*******   READ IN BLOCK OF DATA; INCREMENT COUNTER
C*******
   45 CONTINUE
c        READ(KRIES,425,END=900) (TT(IN),TTDTID(IN),TID(IN),PRS(IN)
c       @ TMP(IN),AMB(IN),TIME(IN),IN=1,NGO)
do 16 in=1,ngo
read(kries,425,end=900)tt(in),ttdtid(in),tid(in),time(in)
16 continue
  425 FORMAT(4E15.7)
   50 CONTINUE
        NLEFT = NLEFT - NGO
C*******
C*******   PROCESSING LOOP TO BE REPEATED FOR EACH SAMPLING PERIOD
C*******
        DO 80 I=1,NGO
        NTOTL = NTOTL + 1
        YRHR=TIME(I)
        DYRHR=DBLE(YRHR)
C*******
C*******   IF IT IS THE FIRST TIME THROUGH LOOP,
C*******   SEARCH TO FIND MONTH WHICH INCLUDES CURRENT YEARHOUR
C*******
        IF (NTOTL .EQ. 1) THEN
          DO 70 II=1,24
          DEL = DYRHR-DBLE(MNTH(II))
          IF (DEL .LE. 0) THEN
           LOW = II-1
           HI=II
           DHMNTH=MNTH(HI)
           DLMNTH=MNTH(LOW)
           TDIF = LSTDEL
           GO TO 75
          ELSE IF (II .LT. 24) THEN
           LSTDEL=DEL
          ELSE
           LOW = II
           HI=25
           DLMNTH=MNTH(LOW)
           DHMNTH=YREND(NOYRS)
```

```
          TDIF = DEL
          GO TO 75
         END IF
   70   CONTINUE
   75   CONTINUE
       ELSE
         DEL=DYRHR-DHMNTH
         IF (DEL .GT. 0.0) THEN
           LOW=LOW+1
           IF(LOW .GT. 24) GO TO 999
           DLMNTH=MNTH(LOW)
           HI=HI+1
           IF(HI .LE. 24) THEN
            DHMNTH=DBLE(MNTH(HI))
           ELSE
            DHMNTH=YREND(NOYRS)
           END IF
         END IF
         TDIF=DYRHR-DLMNTH
       END IF
C*******
C*******   INTERPOLATE TO CALCULATE SEASONAL CORRECTION FACTOR
C*******
       SCF= DBLE(CF(LOW)) + TDIF*DBLE(DSLOPE(LOW))
       TTSCF(I)= TT(I) + SCF
       TDTSCF(I)=TTDTID(I) + SCF
   80 CONTINUE
C*******
C*******   WRITE BLOCK OF CORRECTED DATA TO DISK.  IF END OF DATA,
C*******   WRITE WRAP-UP INFO TO USER'S LOG.
C*******
       WRITE(KWIES,425) (TTSCF(II),TDTSCF(II),TID(II),TIME(II),II=1,NGO)
       IF(NFLAG  .EQ. -1) GO TO 899
       IF (NLEFT .LT. NGO) NGO = NLEFT
       IF (NTOTL .LT. NPTS) GO TO 45
  899 WRITE(KWLOG,430) NTOTL
  430 FORMAT('0',I5,' CORRECTED TRAVEL TIMES WERE WRITTEN TO DISK')
       WRITE(*,43)
       STOP
C*******
C*******  ERROR CONDITIONS:
C*******  1) END OF DATA ENCOUNTERED UNEXPECTEDLY
C*******
  900 CONTINUE
       NFLAG=-1
       WRITE(KWLOG,436)
  436 FORMAT('  FEWER POINTS THAN EXPECTED ON INPUT'/
     @       '  NTOTL WILL BE REVISED')
       NGO = IN - 1
       GO TO 50
C*******
C*******  2) DATASET SPANS MORE THAN TWO CALENDAR YEARS
C*******
  999 CONTINUE
```

```
      WRITE(KWLOG,435)
  435 FORMAT('  *****  RUN TERMINATED  *****'/
     C       '  DATASET EXCEEDED 24 MONTHS')
      WRITE(*,43)
 43   FORMAT(1X,//,5X,' YOUR TEMINAL CONTROL HAS NOW BEEN RETURNED.')
      STOP
      END
```

## 3.8 RESPO_JUL88.FOR

RESPO_JUL88 is linked with subroutines and functions from the library 'VAX_TIDELIB'. RESPO_JUL88 and some of the library routines (POTTY, WEIGHTY, SPONTY, TADM, and HG) are listed here.

```
C      THIS PROGRAM DOES TIDAL RESPONSE ANALYSIS (SEE MUNK & CARTWRIGHT,
C      SUBROUTINES AND FUNCTIONS FROM 'TIDELIB' ARE USED.
C      IMPLEMENTED ON THE PRIME BY DAVID LAI 10/20/80.
C         NO RESTRICTIONS ON THE length OF SERIES (PRIME HAS VIRTUAL
C         MEMORY). COMPUTATIONS ARE DONE ONLY ONCE THROUGH THE WHOLE SERIE
C         I.E. SERIES NOT CUT INTO SEGMENTS.
C         START TIME (SYY), length OF SERIES (length), DELTA TIME (D)
C         NEED TO BE PUT IN.
C         DATA READ STATEMENTS REQUIRED CHANGES ACCORDING TO DATASET.
C         OPTIONS ARE SET BY DATA STATEMENTS:
C         *** FOLLOWING VALUES .EQ. 1 INDICATE::
C      IADJD -- SUBTRACT MEAN FROM SERIES AND MULTIPLY BY CALIB.
C      IDADM -- COMPUTE ADMITTANCE
C      IDRSP -- CREATE PREDICTED TIDE SERIES
C      IRESID - COMPUTE RESIDUAL SERIES
C         *** FOLLOWING VALUE .EQ.0 INDICATES ::
C      KOMPLX -- ONLY REAL PART OF PREDICTED SERIES IS TO BE RETAINED.
C         *** FOLLOWING VALUES .LE.0 INDICATE::
C      IPRIND -- PRINT PARTIAL LIST OF NORMALIZED INPUT DATA
C      IPRINP -- PRINT PARTIAL LIST OF PREDICTED TIDE
C      IPRINR -- PRINT PARTIAL LIST OF RESIDUAL SERIES
C                    **** OTHERWISE THE WHOLE SERIES ARE PRINTED ***
C         *** FOLLOWING QUANTITIES DEPEND ON TYPE OF TIDES INVOLVED.
C         *** SEE LISTINGS OF TIDELIB SUBROUTINES FOR DETAILS ****
C      LGAMMA, NUMGMN, MORDER, NDEGRE, JP, KP, HH, KH, NP, NPHGP1, NH.
C*************************************************
C
C     XXM1 DIMENSIONED AT LEAST  4*NPH*(NPH+1)  WHERE NPH IS THE NUMBER
C     OF  P,H  COMBINATIONS.
C
C     C DIMENSIONED ATLEAST (2*length)+1000
C
C     Y DIMENSIONED (length)   WHERE length IS length OF TIME SERIES
C
C     YPRED DIMENSIONED (length)
C
C         I/O FILES/PARAMETERS
C      ==========================
C*****  KCTRL (UNIT 13) - CONTROL FILE
C*****  KOUT  (UNIT 15) - RESPO OUTPUT
C*****  KIN   (UNIT 16) - DESPIKED INPUT
C
C=================================================================
       parameter ( n_dim1 = 25000 )
       parameter ( n_dim2 = 2 * n_dim1 + 1000 )
       PARAMETER(KCTRL=13,KOUT=15,KIN=16)
       COMPLEX C(n_dim2)
       COMMON /DUM1/ C
       COMMON /DUM2/ Y(n_dim1)
```

```
      COMMON /DUM3/ YPRED(n_dim1)
      COMMON /DUM4/ XXM1(728)  ! nph = 13
      COMMON /DUM5/ YRHR(n_dim1)
      COMMON /DUM6/ YRES(n_dim1)
C*****************************************************
      COMMON /RESTOR/ INISHR,SXR,EXR,SYR,EYR,DR,LIMPR,KOMPLX
      COMMON /TEAPOT/ LEQPOT,LF,THETAO,PHIO,SD,DD,ED,INISHT
      COMMON /WAITER/ INISHL,IFINAL,SX,EX,SY,EY,D,LIMP
      DOUBLE PRECISION YSUM
      DOUBLE PRECISION SD,DD,ED,SYY,EYY,yearhr
      DOUBLE PRECISION XXM
      DIMENSION LGAMMA(10),MORDER(10),NDEGRE(10),PHWTS(80),HH(10),
     + JP(10),H(10),NP(10),NH(10)
      DIMENSION ORAY(20),NCONST(10)
      DIMENSION ITITLD(4),ITITLP(4),ITITLR(4)
      DIMENSION XXM(1)
      CHARACTER*40 HEADR,FORM
      REAL *8 ANAME(5),SAM
      integer leap,spread
      EQUIVALENCE(XXM1(1),XXM(1))
      NAMELIST/CARD1/HEADR
      NAMELIST/CARD2/FORM
      NAMELIST/CARD3/length,year,yearhr,D
C*******************************************************************
      DATA ANAME/'NYY-1   ','POTTY   ','WEIHTY  ','TADM    ','SPONTY  '/
      DATA ITITLD/4H1NOR,4HMALI,4HZED ,4HDATA/
      DATA ITITLP/4H1PRE,4HDICT,4HED T,4HIDE /
      DATA ITITLR/4HOTID,4HE RE,4HSIDU,4HAL  /
      DATA IDWTS/ 1/
C     IADJD.NE.0 MEANS REMOVE MEAN FROM SERIES AND MULTIPLY BY CALIB.
      DATA IADJD/1/, CALIB/1.0/, IPRIND/-1/
      DATA IDADM/1/
      DATA IDRSP/1/ , KOMPLX/0/ , IPRINP/-1/
      DATA IRESID/1/ , IPRINR/-1/ , RSCALE/1E3/
      DATA LGAMMA /3,3, 8*0/ , NUMGMN/2/
      DATA MORDER /1,2, 8*0/
      DATA NDEGRE /2,2, 8*0/
      DATA JP /1,2, 8*0/ , KP/2/
      DATA HH / -48. , 0. , 48., 7*0.0 / , KH/3/
C     DATA HH /-96.,-48.,0.,48.,96.,5*0.0/, KH/5/
      DATA NP/0,2, 8*0/ , NPHGP1/2/
      DATA NH/0,3, 8*0/
C     DATA NH/0,5,8*0/
C********************************************************************
      HCORR(HHH)=FLOAT(IROUND(HHH/D))*D
! 900  FORMAT('0',F9.6,11F10.6)
  900 FORMAT('0',6f15.6)
  90  FORMAT(12F6.2)
  91  FORMAT(4A4//(6(F10.3,F8.3)))
  81  FORMAT('OSYY= ',D20.10,' length=',I10,' D=',D20.10,' EYY=',D20.10)
  910 FORMAT(9H1ISKIPR =,I6)
    2 FORMAT(3HO   ,A6,F6.0)
  443 FORMAT(8HOPHWTS =)
  444 FORMAT(F6.0,F8.2,2F11.6)
```

```
C*********************************************************************
C      *** INSERT SYY=START TIME IN HOURS FROM BEGINNING OF THIS CENTURY
C      ***         length=length OF SERIES
C      ***         D=DELTA TIME IN HOURS
       open(UNIT=kctrl,status='old',FORM='FORMATTED',READONLY)
       read(kctrl,NML=CARD1)
       read(kctrl,NML=CARD2)
       READ(KCTRL,NML=CARD3)
C*****
C***** Conversion from year & year hour to Year Hour from 1900
C*****
Cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C***** correction to conversion.  Spread was previously defined as
C*****   spread=year-1900
C***** The subtraction of unity insures the proper treatment of a leap
C***** year.  This was verified with Dr Wimbush's Kalday function.
C*****
C***** A negative yearhr error trap was removed.  Karen and I were
C***** able to justify why positive year hours were necessary.
C*****
C*****     E.Fields 9-Feb-90
Cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       spread=year-1900-1
       leap=int(spread/4.)
       syy=(spread+1)*8760+float(leap)*24+yearhr
       print *,HEADR,FORM
       print *,length,year,yearhr
C       if (yearhr.LT.0.0) then
C         print 41,headr
C 41      format(1x,a40,/,5x,
C      @          'error in input yearhr. It must be positive!')
C         call exit
C       endif
C      ******* END OF INPUT
C*********************************************************
       EYY=SYY+DBLE(FLOAT(length-1))*D
       PRINT 81,SYY,length,D,EYY
       H(1)=HCORR(HH(1))
       HMIN=H(1)
       HMAX=H(1)
       DO 820 I=2,KH
         H(I)=HCORR(HH(I))
         IF(H(I).LT.HMIN) HMIN=H(I)
         IF(H(I).GT.HMAX) HMAX=H(I)
 820   CONTINUE
C
C      *** TIDPOT ***
       LEQPOT=1
       LF=0
       THETA0=0.
       PHI0=0.
       SD=SYY+DBLE(HMIN)
       DD=D
       ED=EYY+DBLE(HMAX)
```

```
      INISHT=1
      TRY=POTTY(LGAMMA,MORDER,NDEGRE,NUMGMN, C,NTIM)
      SAM=ANAME(2)
      PRINT_2,SAM,TRY
      IF(TRY.EQ.0.) GO TO 40
      CALL EXIT
   40 CONTINUE
      PRINT 769, NTIM
  769 FORMAT('0NTIM =',I6)
      PRINT 336, (C(I), I=1,12)
!      PRINT 336, (C(I), I=1,200)
  336 FORMAT(27H0-TIDPOT SERIES BEGINNING -,(/4(5X,2F12.6)))
C
C     *** TIDWTS ***
C
C     *** CHECK TO READ IN WEIGHTS ***
      IF(IDWTS.GT.0) GO TO 48
      KPS=1
      KPHWTS=4*MAX0(KPS,-IDWTS)
      READ 444, (PHWTS(I),I=1,KPHWTS)
   48 INISHL=1
      IFINAL=1
      SY=SYY
      EY=EYY
      D=DD
      SX=SD
      EX=ED
      LIMP=NUMGMN
      IF(IDWTS.LE.0 .AND. (IDRSP.LE.0 .OR. IRESID.LE.0)) GO TO 106
C***************************************************************
C
C
C     ***** READ IN DATA SERIES ********
C
      open(UNIT=kin,status='old',FORM='FORMATTED',READONLY)
      READ(KIN,FORM) (Y(I), yrhr(i), I=1,length)
!      do nn = 1 , length
!        y(nn) = y(nn) * 100.
!      enddo
C
C
C     ***** END OF READ DATA  *********
C
C**************************************************************
      NY=INT(1.5+(EY-SY)/D)
      PRINT 770, NY
  770 FORMAT('0NY   =',I6)
      PRINT 900, (Y(I), I=1,12)
      NYM11=NY-11
      PRINT 900, (Y(I), I=NYM11,NY)
      YSUM=0.
      DO 70 I=1,NY
        YSUM=YSUM+DBLE(Y(I))
   70 continue
```

```
      YAVE=YSUM/DBLE(FLOAT(NY))
      PRINT 771, YAVE
  771 FORMAT('0YAVE =',E13.7)
      KUP=MIN0(NY,500)
      IF(IADJD.LE.0) GO TO 95
C        REMOVE MEAN AND NORMALIZE
      PRINT 775, CALIB
  775 FORMAT('0CALIB =',E12.4)
      DO 80 I=1,NY
        Y(I)=(Y(I)-YAVE)*DBLE(CALIB)
   80 continue
      NNY=NY
      IF(IPRIND .LE. 0) NNY=KUP
      PRINT 91, ITITLD,(Y(I), I=1,NNY)
 95   IF(IDWTS.LE.0) GO TO 106
          PRINT 8006, (NP(I), I=1,NPHGP1)
 8006     FORMAT('0  NP',30I4)
      TRY=WEIHTY(C, Y, JP,KP, H,KH, NP,NH,NPHGP1, XXM1,XXM,
     + PHWTS,KPHWTS)
      SAM=ANAME(3)
      PRINT 2,SAM,TRY
      IF(TRY.EQ.0.) GO TO 100
      CALL EXIT
  100 INISHL=0
  106 PRINT 443
      PRINT 444, (PHWTS(I), I=1,KPHWTS)
C
C     *** TIDADM ***
      IF(IDADM .LE. 0) GO TO 220
      KEEP=1
      DO 200 IC=1,NUMGMN
        CMPNT=FLOAT(IC)
        IF(LGAMMA(IC).EQ.4) GO TO 190
        IF(NDEGRE(IC).GT.2) GO TO 3000
        IF(MORDER(IC).EQ.0) GO TO 2500
        DELF=0.0366011
        IF(MORDER(IC).EQ.2) GO TO 2000
        FS=0.8929346
        NCONST(1)=2
        NCONST(2)=4
        NNC=2
  165   NTYPE=10
        LO=10
        TRY=TADM(PHWTS,KPHWTS, ORAY,LO, FS,DELF, CMPNT,KEEP)
        CALL HG(ORAY,LO, NCONST,NNC, MORDER(IC), NTYPE)
        SAM=ANAME(4)
        PRINT 2,SAM,TRY
        IF(TRY.EQ.0.) GO TO 170
        CALL EXIT
  170   DELF=0.104018
        FS=0.8932441
        IF(MORDER(IC).EQ.2) FS=1.895982
        NTYPE=10
  175   LO=4
```

```
  180   TRY=TADM(PHWTS,KPHWTS, ORAY,LO, FS,DELF, CMPNT,KEEP)
        NCONST(1)=1
        NCONST(2)=2
        NNC=2
        CALL HG(ORAY,LO, NCONST,NNC, MORDER(IC), NTYPE)
        SAM=ANAME(4)
        PRINT 2,SAM,TRY
        IF(TRY.EQ.0.) GO TO 200
        CALL EXIT
  190   DELF=0.005475819
        FS=0.9972620907
        IF(NDEGRE(IC).EQ.1) FS=1.0
        IF(MORDER(IC).EQ.2) FS=2.0
        NTYPE=-IFIX(2.*ABS(1.-FS)+0.999)
        LO=4
        GO TO 180
 2000   FS=1.8590714
        NCONST(1)=3
        NCONST(2)=5
        NNC=2
        GO TO 165
 2500   FS=0.00547582
        DELF=0.06772639
        NTYPE=-10
        GO TO 175
 3000   IF(MORDER(IC).NE.3) GO TO 3300
        DELF=0.036291647
        FS=2.862118775
  3200  NTYPE=IROUND(FS)
        LO=4
        GO TO 180
  3300  DELF=0.073202204
        FS=0.9664462631
        IF(MORDER(IC).EQ.2) FS=1.895672514
        GO TO 3200
  200 CONTINUE
          PRINT 8006, (NP(I), I=1,NPHGP1)
C
C     *** TIDRSP ***
  220 IF(IDRSP.LE.0) GO TO 160
      LIMPR=LIMP
      NP(2)=KP
C     ***** NP(2)=1 -- PREDICTED TIDES CONSIST OF ONLY SEMI-DIURNAL **
C     NP(2)=1
C     *****
      NPGP1=2
      INISHR=1
      SXR=SX
      EXR=EX
      SYR=SY
      EYR=EY
      DR=D
      TRY=SPONTY(C, PHWTS,KPHWTS, JP, NP,NPGP1, YPRED,KY)
C     **** FOLLOWING STATEMENT COMPUTES ONLY SEMI-DIURNAL TIDES ***
```

```
C      TRY=SPONTY(C, PHWTS,KPHWTS, 2, NP,2, YPRED,KY)
       SAM=ANAME(5)
       PRINT 2,SAM,TRY
       IF(TRY.EQ.0.) GO TO 130
       CALL EXIT   -
  130  CONTINUE
       PRINT 772, KY
  772  FORMAT('OKY   =',I6)
       KOMP=MINO(2,IABS(KOMPLX)+1)
       KYY=KOMP*KY
       KUP=MINO(KY,500)
       NNY=KY
       IF(IPRINP .LE. 0) NNY=KUP
       NNY=KOMP*NNY
       PRINT 91, ITITLP,(YPRED(I), I=1,NNY)
  150  IF(IRESID .LE. 0) GO TO 160
       IF(KOMPLX .NE. 0) GO TO 153
       DO 152 I=1,KY
         YRES(I)=Y(I)-YPRED(I)
  152  continue
       GO TO 156
  153  DO 154 I=1,KY
         YRES(I)=Y(I)-YPRED(2*I-1)
  154  continue
  156  NNY=KY
       IF(IPRINR .LE. 0) NNY=KUP
       ISKIPR=MAXO(1,IPRINR)
       PRINT 910, ISKIPR
       PRINT 91,ITITLR,(YRES(I), I=1,NNY,ISKIPR)
  160  CONTINUE
C*******************************************************************
C
C      ***  WRITE RESIDUAL TIME SERIES  ****
C
       OPEN(UNIT=KOUT,status='new',FORM='FORMATTED')
       do 13 i=1, ky
       WRITE(KOUT,162) y(i),yres(i),ypred(i),yrhr(i)
   13  continue
  162  FORMAT(4e15.7)
       CLOSE(KIN)
       close(kctrl)
       CLOSE(KOUT)
C
C*****************************************************************
       CALL EXIT
       END
       FUNCTION POTTY(LGAMMA,MORDER,NDEGRE,NUMGMN, C,NTIM)

C
C TITLE - POTTY = POTENTIAL TYDE
C      GENERATION OF TIDE POTENTIALS
C
C
C                       ---ABSTRACT---
```

```
C
C              IF  LEQPOT .NE. 0  POTTY(LGAMMA,MORDER,NDEGRE,NUMGMN,
C              C,NTIM)  GENERATES FUNCTIONS RELATED TO THE TIDE
C              POTENTIALS.  THE FUNCTIONS ARE NOT COMPUTED AS A SUPER-
C              POSITION OF TIME HARMONICS IN THE CLASSICAL SENSE, BUT
C              DIRECTLY FROM THE KNOWN ORBITAL CONSTANTS.
C              (SEE W.H. MUNK AND D.E. CARTWRIGHT, 1966. TIDAL SPECTRO-
C              SCOPY AND PREDICTION, PHIL. TRANS. ROY. SOC. A, 259,
C              533-581)
C
C                  IF  LEQPOT=0  POTTY(LGAMMA,MORDER,NDEGRE,NUMGMN,C)
C              GENERATES THE  GAMMA EQUILIBRIUM TIDE AT A
C              PLACE  THETA,PHI  (IN DEGREES) AS DERIVED FROM THE FUNDA-
C              MENTAL DEFINITIONS WITHOUT EXPANSION INTO SPHERICAL
C              HARMONICS.
C
C
C                     --STATISTICS--
C
C LANGUAGE   - FORTRAN IV (CDC3600,B6500)
C EQUIPMENT  - NO SPECIAL REQUIREMENTS
C STORAGE    - 610(OCTAL) = 392(DECIMAL) LOCATIONS
C SPEED      -
C AUTHOR     - MARK WIMBUSH    IGPP    JUL 1970
C LAST MOD   - MARK WIMBUSH    NOVA    APR 1972
C CATAGORIES
C STATUS     -
C
C LIBRARY ROUTINES USED - AMENPI, RECURQ, SHMIDT, SETUPM, ORBITS
C SYSTEM ROUTINES USED - Q2Q07110,Q1Q00310,Q1Q04310,Q1Q04330,Q1Q02330,
C              DMOD,XINTF,SQRTF,SINF,COSF,AIMAG
C
C
C                     ----USAGE----
C
C SAMPLE CALL
C     J = POTTY(LGAMMA,MORDER,NDEGRE,NUMGMN, C,NTIM)
C
C NOTE       - DIMENSION OF ARRAY  C  IS  NUMGMN*NTIM
C              (COMPLEX UNLESS  LEQPOT=0)
C
C
C INPUTS
C
C     LGAMMA(I)  =1 FOR MOON@S GRAVITATIONAL POTENTIAL
C                =2 FOR SUN@S GRAVITATIONAL POTENTIAL
C                =3 FOR TOTAL GRAVITATIONAL POTENTIAL
C                =4 FOR SUN@S RADIATIONAL POTENTIAL
C
C     MORDER(I)  VALUE OF  M  IN  ITH  TRIPLET
C              (ORDER OF SPHERICAL HARMONIC)
C              NOT MEANINGFUL IF  LEQPOT=0
C
C     NDEGRE(I)  VALUE OF  N  IN  ITH  TRIPLET
```

```
C              (DEGREE OF SPHERICAL HARMONIC)
C              NOT MEANINGFUL IF  LEQPOT=0
C
C     NUMGMN   NUMBER OF  GAMMA,M,N  TRIPLETS
C
C ----COMMON /TEAPOT/
C
C     LEQPOT   =0  IF THE EQUILIBRIUM TIDE IS TO BE STORED AT  C.
C              LF  AND ARRAYS  MORDER  AND  NDEGRE  ARE NOT MEANINGFUL
C              AND ARE NOT CHECKED.
C              .NE. 0  IF TIDE POTENTIALS ARE TO BE STORED AT  C.
C
C     LF       =0  IF  THETAO AND  PHIO  ARE TO BE IGNORED
C              NOT MEANINGFUL IF  LEQPOT=0
C
C     THETAO   LOCAL COLATITUDE, IN DEGREES
C              (NEEDED ONLY IF  LF .NE. 0  OR IF LEQPOT=0)
C
C     PHIO     GREENWICH EAST LONGITUDE, IN DEGREES
C              (NEEDED ONLY IF  LF .NE. 0  OR IF LEQPOT=0)
C
C     SD       START TIME IN HOURS (DOUBLE PRECISION)
C
C     DD       INCREMENT TIME IN HOURS (DOUBLE PRECISION)
C
C     ED       END TIME IN HOURS (DOUBLE PRECISION)
C
C              NOTE. ZERO TIME IS TAKEN TO BE  1900 JAN 1 0000HRS GMT
C
C     INISHL   MUST BE SET NON-ZERO IN INITIAL CALL OF  POTTY.
C              IN FURTHER CALLS,  INISHL  MAY BE SET TO ZERO IF THE ONLY
C              OTHER CHANGED INPUT PARAMETERS ARE  SD, DD, AND  ED
C
C
C OUTPUTS
C
C     POTTY    =0. IF INPUT ITEMS HAVE VALID VALUES
C              =1. IF  MORDER .GT. NDEGRE
C              =2. IF  NDEGRE .LT. 1
C              =3. IF  MORDER .LT. 0
C              =4. IF  LGAMMA .LT. 1  OR  LGAMMA .GT. 4
C              =5. IF  NUMGMN .LT. 1
C              =6. IF  IMPOSSIBLE COLATITUDE (THEATO)
C              =7. IF  INVALID TIME GROUP (SD,DD,ED)
C
C     C        IF LEQPOT .NE. 0, THE MERGED COMPLEX SERIES  C  OR  F  OF
C              TIDE POTENTIALS ACCORDING AS  LF=0  OR  1  (SEE EQN. A7
C              OF MUNK AND CARTWRIGHT OR DESCRIPTION OF  BOOM  STATEMENT
C              TIDPOT)
C              IF LEQPOT = 0, THE GAMMA EQUILIBRIUM TIDE  F(THETAO,PHIO)
C              AS DERIVED FROM THE FUNDAMENTAL DEFINITIONS WITHOUT
C              EXPANSION INTO SPHERICAL HARMONICS.
C
C     NTIM     NUMBER OF DIFFERENT TIMES FOR WHICH TERMS OF  C  ARE
```

```
C               COMPUTED  -  I.E. INT(1.5+(ED-SD)/DD)
C
C
C EXAMPLES
C
C
C PROGRAM FOLLOWS BELOW
C

!      implicit real*8 (a-h,o-z)

      COMMON /TEAPOT/ LEQPOT,LF,THETA0,PHI0,SD,DD,ED,INISHL
      COMMON /WORKIN/ XNPI(138),RADN(16),DEIMLS(17),DEIMLM(17),
     1 DEIMLF(17),REQ(285),QS(153),QM(153),QF(153)
      COMMON /POTLUK/ LAST,F,CTHETA,STHETA,CZ(2),SZ(2),BIGL(2),RBDR(2),
     1 TD,HR
c\\\\\\\\\\\\
      real*8 SD,DD,ED,TD,TGP12D
!      real*8 pax,amdme,pid180
      COMPLEX DEIMLS,DEIMLM,DEIMLF,DK,CK
      DIMENSION LGAMMA(NUMGMN),MORDER(NUMGMN),NDEGRE(NUMGMN),
     1 C(1),PAX(2),AMDME(2)
C         (SEE ●NOTE● ABOVE FOR TRUE DIMENSION OF  C)
      EQUIVALENCE (IGMN,XNPI(1))
C    1964 I.A.U. ASTRONOMICAL CONSTANTS (SEE AMERICAN EPHEMERIS AND
C   NAUTICAL ALMANAC 1971, P.481)
C    SOLAR PARALLAX = 8●●.794
C   (8.794/3600.)*(PI/180.)=4.2634515117E-5
C    SINE PARALLAX FOR MOON = 3422●●.451
C   (3422.451/3600.)*(PI/180.)=1.6592510677E-2
C    AEQUATORIAL = 6378160 M
C    MSUN/(MEARTH+MMOON) = 328912 ,  MEARTH/MMOON = 81.30
C   (6378160.*100.CM)*328912.*(1.+1./81.30)=2.1236572163E14 CM
C   (6378160.*100.CM)/81.30=7.8452152522E6 CM
      DATA PAX /4.2634515117e-5,   1.6592510677e-2/,
     1 AMDME /2.1236572163e14,  7.8452152522e6/,
     2 PID180 /.01745329252e0/

C .  POTTY IS THE MAIN INDEPENDENT SUBPROGRAMME FOR STATEMENTS
C   ●TIDPOT● AND ●TIDEQU●.   (PROGRAMME AND ASSOCIATED SUBROUTINES
C   WRITTEN BY MARK WIMBUSH - JULY 1970)

      POTTY=0.
      IF(SD.NE.ED) GO TO 10
      NTIM=1
      GO TO 30
   10 DDEMSD=SNGL(DD/(ED-SD))
      IF(DDEMSD.GT.0.) GO TO 20
      POTTY=7.
      GO TO 40
   20 NTIM=INT(1./DDEMSD+1.5)
C    SKIP NEXT SECTION IF NOT INITIAL CALL
   30 IF(INISHL.EQ.0) GO TO 70
      PHI=PHI0*PID180
```

```
      THETA=THETA0*PID180
      CTHETA=COS(THETA)
      STHETA=SIN(THETA)
      IF(NUMGMN.LT.1) POTTY=5.
      IF(STHETA.LT.0..AND.(LEQPOT.EQ.0.OR.LF.NE.0)) POTTY=6.
   40 IF(POTTY.NE.0.) RETURN
C     COMPUTE  NMAX (MAXIMUM N),  NRMAX (MAXIMUM RADIATIONAL N),
C    FORM  LAST  ACCORDING TO DESCRIPTION ABOVE, AND CHECK FOR INVALID
C    GAMMA,M,N  TRIPLETS (ASSIGNING VALUES TO POTTY ACCORDING TO
C    DESCRIPTION ABOVE)
      NMAX=0
      NRMAX=0
      LAST=0
      MAST=-1

      DO 60 IGMN=1,NUMGMN
        LG=LGAMMA(IGMN)
        M=MORDER(IGMN)
        N=NDEGRE(IGMN)
        IF(LEQPOT.EQ.0) GO TO 50
        IF(N.GT.NMAX) NMAX=N
        IF(LG.EQ.4.AND.N.GT.NRMAX) NRMAX=N
        IF(M.GT.N) POTTY=1.
        IF(N.LT.1) POTTY=2.
        IF(M.LT.0) POTTY=3.
   50   IF(LG.LT.1.OR.LG.GT.4) POTTY=4.
C     IF BAD  GAMMA,M,N  TRIPLET IS FOUND,  RETURN
        IF(POTTY.NE.0.) RETURN
        IF(LG.NE.1) MAST=1
        IF(LG.NE.2.AND.LG.NE.4) LAST=1
   60 CONTINUE

      LAST=LAST*MAST
      NMAXP1=NMAX+1

C---  SKIP NEXT SECTION IF STATEMENT IS @TIDEQU@

      IF(LEQPOT.EQ.0) GO TO 70

C---  CALCULATE TIME INDEPENDENT FACTORS, ALSO CALCULATE COEFFICIENTS
C---  FOR RECURSION FORMULA USED TO GENERATE SCHMIDT FUNCTION PART
C---  OF SPHERICAL HARMONICS

      CALL ANENPI(PAX,AMDME, LGAMMA,NDEGRE,NUMGMN, NRMAX)
      CALL RECURQ(NMAX)
      F=0.

C---  SKIP NEXT SECTION IF @F@ IS NOT SET

      IF(LF.EQ.0) GO TO 70

C---  COMPUTE COMPONENTS OF SPHERICAL HARMONICS AT THE LOCATION
C---  THETA,PHI

```

```
      CALL SHMIDT(NMAX,CTHETA,STHETA, QF)
      F=1.
      CALL SETUPM(PHI, DEIMLF,NMAXP1)
      F=-F
   70 K=1

C---  SD(DD)ED ARE START(INTERVAL)END TIMES IN HOURS SINCE
C---  1900 JAN 1 0000 HRS GMT

      TGP12D=SD+12D0
      DO 260 ITIM=1,NTIM
        HR=SNGL(DMOD(TGP12D,24D0))
C---    TD IS TIME IN JULIAN CENTURIES SINCE 1899 DEC 31 NOON GMT
        TD=TGP12D/876600D0
C---    CALCULATE NEEDED ORBITAL PARAMETERS OF SUN AND MOON
        CALL ORBITS
C---    BRANCH IF STATEMENT IS @TIDEQU@
        IF(LEQPOT.EQ.0) GO TO 200
c\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
C---    SKIP NEXT SECTION IF ALL GAMMAS ARE @MG@
        IF(LAST.EQ.-1) GO TO 80
C---    COMPUTE SPHERICAL HARMONIC COMPONENTS OF GREENWICH COORDINATES
C---    OF SUN
        CALL SHMIDT(NMAX,CZ(1),SZ(1), QS)
        CALL SETUPM(BIGL(1), DEIMLS,NMAXP1)
C---    SKIP NEXT SECTION IF ALL GAMMAS ARE @SG@ OR @RAD@
        IF(LAST.EQ.0) GO TO 90

C---    COMPUTE SPHERICAL HARMONIC COMPONENTS OF GREENWICH COORDINATES
C---    OF MOON

   80   CALL SHMIDT(NMAX,CZ(2),SZ(2), QM)

        CALL SETUPM(BIGL(2), DEIMLM,NMAXP1)
c\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
   90   II=1
        DO 190 I=1,NUMGMN
          MP1=MORDER(I)+1
          NP1=NDEGRE(I)+1
          J=NP1*(NP1-1)/2+MP1
          LG=LGAMMA(I)
          CK=(0.,0.)
  100     XK=XNPI(II)
          IF(XK.EQ.0.) GO TO 180
          GO TO (160,130,120,110,150), LG
  110     RK=RBDR(1)**2
          GO TO 140
  120     LG=5
  130     RK=RBDR(1)**NP1
  140     QK=QS(J)
          DK=DEIMLS(MP1)
          GO TO 170
  150     LG=0
  160     RK=RBDR(2)**NP1
```

```
              QK=QM(J)
              DK=DEIMLM(MP1)
C---          COMPUTE  C = A + IB
 170          CK=CK+XK*RK*QK*DK
 180          II=II+1 -
              IF(LG.EQ.5) GO TO 100
C---          IF 'F' IS SET MULTIPLY BY VALUE OF SPHERICAL HARMONIC AT
C---          LOCATION  THETA,PHI
              IF(LF.NE.0) CK=QF(J)*DEIMLF(MP1)*CK
              C(2*K-1)=REAL(CK)
              C(2*K)=AIMAG(CK)
              IF(MP1.EQ.1) C(2*K)=1.
 190        K=K+1
            GO TO 260
c\\\\\\\\\\\\\\
C    STATEMENT IS 'TIDEQU'  -  COMPUTE EQUILIBRIUM TIDE AT THETA,PHI
 200        DO 250 I=1,NUMGMN
              SFK=0.
              LSM=1
              IF(LGAMMA(I).EQ.1) LSM=2
 210          CALFA=CTHETA*CZ(LSM)+STHETA*SZ(LSM)*COS(PHI-BIGL(LSM))
              IF(LGAMMA(I).EQ.4) GO TO 220
              P=PAX(LSM)*RBDR(LSM)
              IF(LSM.NE.2) GO TO 230
              SFK=SFK+AMDME(2)*P*(1./SQRT((P-2.*CALFA)*P+1.)-1.-P*CALFA)
              GO TO 240
 220          RK=RBDR(1)**2
              SFK=-.25*RK
              IF(CALFA.GT.0.) SFK=SFK+CALFA*RK
              GO TO 240
 230          GAM=(2.*CALFA-P)*P
              SFK=AMDME(1)*P*(((.2734375*GAM+.3125)*GAM+.375)*
     +            GAM**2-.5*P**2)
              IF(LGAMMA(I).NE.3) GO TO 240
              LSM=2
              GO TO 210
 240          C(K)=SFK
 250        K=K+1

C---        IF END TIME NOT REACHED, GO BACK AND DO CALCULATIONS FOR
C---        NEXT TIME STEP

 260    TGP12D = TGP12D + DD

        RETURN

        END
        FUNCTION WEIHTY(X, Y, JP,KP, H,KH, NP,NH,NPHGP1, XXM1,XXM,
       1 PHWTS,KPHWTS)
C
C
C TITLE - WEIHTY = WEIGHTS, TYDE
C       GENERATES OPTIMUM PREDICTION WEIGHTS
C
```

```
C
C                       ---ABSTRACT---
C
C              WEIHTY  COMPUTES COMPLEX WEIGHTS  W(P,H)  SUCH THAT THE
C              REAL PART OF THE SUM OVER ALL SPECIFIED  P,H  COMBINA-
C              TIONS OF  CONJG(W(P,H))*X(P,T+H)  IS AS CLOSE AS POS-
C              SIBLE TO  Y(T)  IN THE LEAST-SQUARES SENSE.  X(P,T)  RE-
C              PRESENTS THE  PTH  COMPONENT OF THE COMPLEX REFERENCE
C              SERIES  X  AT TIME  T.  Y(T) REPRESENTS THE VALUE AT TIME
C              T  OF THE SERIES TO BE PREDICTED. THE  P  AND  H  VALUES
C              TO BE USED ARE GIVEN BY ARRAYS  JP  AND  H,  AND THE  P,H
C              COMBINATIONS ARE SPECIFIED BY ARRAYS  NP  AND  NH  - FOR
C              EACH  N  SUCH THAT  0 .LT. N .LE. NPHGP1,  ALL  JP(J)
C              SUCH THAT NP(N-1) .LT. J .LE. NP(N)  ARE COMBINED WITH
C              ALL  H(I) SUCH THAT NH(N-1) .LT. I .LE. NH(N) .  THE OUT-
C              PUT ARRAY  PHWTS  CONSISTS OF  P,  H, W(P,H)  MERGED
C              (I.E. A 4 REAL COMPONENT SERIES).
C
C
C                       --STATISTICS--
C
C LANGUAGE   - UCSD FORTRAN 63
C EQUIPMENT  - NO SPECIAL REQUIREMENTS
C STORAGE    - 484 WORDS FOR THIS PROGRAMME + 436 WORDS FOR
C              ASSOCIATED SUBPROGRAMMES (MAVDUB,ISMI,INTCH) +
C              1018 WORDS COMMON + SPACE FOR ARRAYS  X,  Y,  JP,  H,
C              NP,  NH,  XXM1,  PHWTS  (I.E. 2*KX+KY+KP+KH+2*NPHGP1+
C              KM1+KPHWTS  WORDS)
C SPEED      -
C AUTHOR     - MARK WIMBUSH    IGPP    JUL 1970
C LAST MOD   - MARK WIMBUSH    NOVA    APR 1972
C CATAGORIES -
C STATUS     -
C
C LIBRARY ROUTINES USED- MAVDUB, ISMI
C SYSTEM ROUTINES USED - IROUND, XINTF, FLOATF, Q8QINGOT, Q8QENGOT,
C              Q8QGOTTY
C
C
C                       ----USAGE----
C
C SAMPLE CALL
C     J = WEIHTY(X,Y,JP,KP,H,KH,NP,NH,NPHGP1,XXM1,XXM, PHWTS,KPHWTS)
C
C NOTE       - XXM1(1),XXM(1) SHOULD BE EQUIVALENCED PRIOR TO
C              CALLING WEIHTY
C            - DIMENSION OF REFERENCE SERIES  X  IS
C              LIMP*INT(1.5+(EX-SX)/D) (COMPLEX)
C            - DIMENSION OF DATA SERIES  Y  IS  KY=INT(1.5+(EY-SY)/D)
C            - DIMENSION OF WORKING STORAGE ARRAY  XXM1  IS
C              4*NPH*(NPH+1),  WHERE  NPH  IS THE TOTAL NUMBER
C              OF  P,H COMBINATIONS SPECIFIED
C            - DIMENSION OF ARRAY  PHWTS  IS  KPHWTS=4*NPH
C
```

```
C
C
C INPUTS
C
C     X(I)      COMPLEX MERGED REFERENCE SERIES ON WHICH THE PREDICTION
C               WEIGHTS ARE TO BE BASED
C
C     Y(I)      SERIES OF OBSERVATIONS FOR WHICH PREDICTION WEIGHTS ARE
C               TO BE FOUND  (ZERO MEAN)
C
C     JP(I)     ARRAY OF  P  VALUES (X  COMPONENT NUMBERS) TO BE USED
C               IN FORMING THE PREDICTION WEIGHTS
C
C     KP        DIMENSION OF ARRAY  JP
C
C     H(I)      ARRAY OF  H  VALUES (X TIME LEADS) TO BE USED IN FORMING
C               THE PREDICTION WEIGHTS
C
C     KH        DIMENSION OF ARRAY  H
C
C     NP(I)     NP(1) = 0
C               NP(I .GT. 1)  IS THE NUMBER OF THE LAST TERM IN THE
C               (I-1)TH  GROUP OF ARRAY  JP
C
C     NH(I)     NH(1) = 0
C               NH(I .GT. 1)  IS THE NUMBER OF THE LAST TERM IN THE
C               (I-1)TH GROUP OF  ARRAY  H
C
C     NPHGP1    DIMENSION OF ARRAY  NP  AND OF ARRAY  NH
C
C     XXM1      AN ARRAY OF WORKING STORAGE NEEDED FOR MATRIX OPERATIONS
C
C ----COMMON /WAITER/
C
C     INISHL    INISHL .NE. 0  INDICATES THAT THIS IS THE INITIAL CALL
C               IN THIS COMPUTATION
C               INISHL .EQ. 0  INDICATES THAT THIS IS NOT THE INITIAL
C               CALL
C
C     IFINAL    IFINAL .NE. 0  INDICATES THAT THE CALL IS THE FINAL CALL
C               IN THIS COMPUTATION.
C               IFINAL = 0  INDICATES THAT THE CALL IS NOT THE FINAL CALL
C
C     SX        START TIME OF SERIES  X
C
C     EX        END TIME OF SERIES  X
C
C     SY        START TIME OF SERIES  Y
C
C     EY        END TIME OF SERIES  Y
C
C     D         UNIFORM TIME INCREMENT OF SERIES  X  AND SERIES  Y
C
C     LIMP      NUMBER OF COMPLEX COMPONENTS MERGED IN SERIES  X
```

```
C
C
C OUTPUTS
C
C       PHWTS(I) ARRAY CONSISTING OF  P VALUE,  H VALUE,  COMPLEX PREDIC-
C                TION WEIGHT W(P,H)  MERGED TOGETHER (I.E. 4 REAL COMPON-
C                ENTS MERGED)
C
C       KPHWTS   DIMENSION OF ARRAY  PHWTS
C
C       WEIHTY   =0.  IF NO ERRORS
C                =1.  IF A  P  VALUE IS GREATER THAN THE NUMBER OF COM-
C                PONENTS IN THE  X  SERIES.
C                =2.  IF A VALUE IN THE ARRAY  NP  EXCEEDS THE DIMENSION
C                OF ARRAY  JP
C                =3.  IF VALUES IN  NP  ARE NOT IN INCREASING ORDER
C                =4.  IF A VALUE IN ARRAY  NH  EXCEEDS THE DIMENSION
C                OF ARRAY  H
C                =5.  IF VALUES IN  NH  ARE NOT IN INCREASING ORDER
C                =6.  X  SERIES STARTS TOO LATE TO ACCOMODATE MINIMUM LEAD
C                =7.  SY  IS NOT ONE OF THE TIMES OF SERIES  X
C                =8.  X  SERIES ENDS TOO EARLY TO ACCOMODATE MAXIMUM LEAD
C                =9.  IF NO  P  VALUES ARE GIVEN
C                =10. IF NO  H  VALUES ARE GIVEN
C                =11. IF  NP(1) .LT. 0
C                =12. IF  NH(1) .LT. 0
C                =13. IF THERE ARE NO  P,H  COMBINATIONS, THAT IS
C                NPHGP1 .LT. 2
C                =14. IF  (TIME INTERVAL)/(END TIME-START TIME)  IS
C                NEGATIVE FOR SERIES  Y
C                =15. IF  (TIME INTERVAL)/(END TIME-START TIME)  IS
C                NEGATIVE FOR SERIES  X
C                =16. IF MATRIX IS SINGULAR
C
C
C EXAMPLES
C
C
C PROGRAM FOLLOWS BELOW
C
      COMMON /WAITER/ ININSHL,IFINAL,SX,EX,SY,EY,D,LIMP
      COMMON /WEIGHT/ JXY,NPH,NPH1,KY,TOT,SIGMA,JI,JK,JD
      COMMON /WORKIN/ JH(125),JPH(125),XX1(250),YXV1(500)
      DOUBLE PRECISION XXM,YXV
      COMPLEX X
      DIMENSION X(1),Y(1),JP(KP),H(KH),NP(NPHGP1),NH(NPHGP1),
     1 XXM1(1),XXM(1),YXV(250),PHWTS(1),PPHG(125)   !!!

C---   (SEE @NOTE@ ABOVE FOR TRUE DIMENSIONS OF  X,Y,XXM1,PHWTS)

      EQUIVALENCE (YXV1(1),YXV(1)), (PPHG(1),JH(1))

      DATA DELTO,DELO/0.8,0.8/

```

```
      WEIHTY=0.
      IF(SX.EQ.EX) GO TO 10
      IF(D/(EX-SX).GT.0.) GO TO 10
      WEIHTY=15.
      RETURN
   10 IF(SY.NE.EY) GO TO 20
      KY=1
      GO TO 40
   20 DDEMSY=D/(EY-SY)
      IF(DDEMSY.GT.0.) GO TO 30
      WEIHTY=14.
      RETURN
   30 KY=INT(1./DDEMSY+1.5)
C          SKIP NEXT SECTION IF NOT INITIAL CALL
   40 IF(INISHL.EQ.0) GO TO 100
c\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
      IF(KP.LT.1) WEIHTY=9.
      IF(KH.LT.1) WEIHTY=10.
      IF(NP(1).LT.0) WEIHTY=11.
      IF(NH(1).LT.0) WEIHTY=12.
      IF(NPHGP1.LT.2) WEIHTY=13.
      IF(WEIHTY.NE.0.) RETURN
c\\\\\\\\\\\\\\\\\\\\\\\\\\\
      DO 50 K=1,KP
        IF(JP(K).GT.LIMP) WEIHTY=1.
   50 CONTINUE
c\\\\\\\\\\\\\
      HMIN=H(1)
      HMAX=H(1)
      DO 60 K=1,KH
        JH(K)=IROUND(H(K)/D)
        IF(H(K).LT.HMIN) HMIN=H(K)
        IF(H(K).GT.HMAX) HMAX=H(K)
   60 CONTINUE
c\\\\\\\\\\\\\
      NPH=0
C          FOR EACH P,H COMBINATION, COMPUTE OFFSET JPH IN SERIES X
      DO 70 IPHGP1=2,NPHGP1
        NP1=NP(IPHGP1-1)+1
        NP2=NP(IPHGP1)
        IF(NP2.GT.KP) WEIHTY=2.
        IF(NP2.LT.NP1) WEIHTY=3.
        NH1=NH(IPHGP1-1)+1
        NH2=NH(IPHGP1)
        IF(NH2.GT.KH) WEIHTY=4.
        IF(NH2.LT.NH1) WEIHTY=5.
        DO 70 IP=NP1,NP2
          JPIP=JP(IP)
          DO 70 IH=NH1,NH2
            NPH=NPH+1
   70 JPH(NPH)=JPIP+JH(IH)*LIMP
c\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
      NPH1=2*NPH
      KPHWTS=2*NPH1
```

```
c\\\\\\\\\\\\\\\\\\\\
 100  XY=SY-SX
      IF(-XY-HMIN.GT.DELTO) WEIHTY=6.
      XY=XY/D
      JXY=IROUND(XY)
      IF(ABS(XY-FLOAT(JXY)).GT.DELO) WEIHTY=7.
      IF(EY+HMAX-EX.GT.DELTO) WEIHTY=8.
      IF(WEIHTY.NE.0.) RETURN
      JXY=JXY*LIMP
C        FORM MATRIX M (UPPER TRIANGULAR) AND VECTOR V (BOTH IN
C        DOUBLE PRECISION)
      CALL MAVDUB(Y,X, XXM)
C        RETURN IF NOT FINAL CALL
      IF(IFINAL.EQ.0) RETURN
c\\\\\\\\\\\\\\\\\\\\\\\\\\\\
C        PACK M AND V IN SINGLE PRECISION FORM

      IJM1=NPH1+1
      IJM2=NPH

      DO 120 JPH1=1,NPH1
        DO 110 IPH1=1,JPH1
          IJM1=IJM1+1
          IJM2=IJM2+1
          XXM1(IJM1)=SNGL(XXM(IJM2))/TOT
 110    continue
        YXV1(JPH1)=SNGL(YXV(JPH1))/TOT
 120  continue

c\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
      SIGMA=SIGMA/TOT
C        SOLVE EQUATION FOR VECTOR OF WEIGHTS W
      IF(ISMI(XXM1).NE.0) GO TO 160
      P=0.
      I=0
      IPHG=0
      DO 150 IPHGP1=2,NPHGP1
        IPHG=IPHG+1
        NP1=NP(IPHGP1-1)+1
        NP2=NP(IPHGP1)
        NH1=NH(IPHGP1-1)+1
        NH2=NH(IPHGP1)
        PRINT 1000, IPHG,(H(IH), IH=NH1,NH2)
        PRINT 1100
        PG=0.
        DO 140 IP=NP1,NP2
          PPG=0.
          I1=I+1
          DO 130 IH=NH1,NH2
            PHWTS(4*I+1)=FLOAT(JP(IP))
            PHWTS(4*I+2)=H(IH)
            PHWTS(4*I+3)=YXV1(2*I+201)
            PHWTS(4*I+4)=YXV1(2*I+202)
            I=I+1
```

```
            PPHG(I)=YXV1(2*I-1)*YXV1(2*I+199)+YXV1(2*I)*YXV1(2*I+200)
 130      PPG=PPG+PPHG(I)
          PRINT 1200, JP(IP),PPG,(PPHG(IH), IH=I1,I)
 140    PG=PG+PPG
        PRINT 1300, PG,IPHG
 150  P=P+PG

      RESIDV=SIGMA-P
      PRINT 1400, SIGMA,P,RESIDV
      I=4*I
      PRINT 1500, (PHWTS(K), K=1,I)
      PRINT 1600
      RETURN
 160  WEIHTY=16.
      RETURN

c\\\\\\\\\\\\

 1000 FORMAT (40H0 PREDICTED VARIANCE MATRIX OF P,H GROUP,I3,3H IS/1H0,
     15X,12HROW SUM    H,F11.4,7F12.4/(17X,8F12.4))
 1100 FORMAT (3H  P)
 1200 FORMAT (1X,I2,2E13.4,7E12.4/(17X,8E12.4))
 1300 FORMAT (5X,11H-----------/4X,E12.4,42H  IS TOTAL PREDICTED VARIANC
     1E OF P,H GROUP,I3/)
 1400 FORMAT (1H0,5X,21HRECORDED  VARIANCE  =,E14.6/6X,21HPREDICTED VARI
     1ANCE  =,E14.6/6X,21HRESIDUAL  VARIANCE  =,E14.6//)
 1500 FORMAT (21H0 GENERATED SERIES IS/3H0 P,6X,1HH,8X,
     + 26HREAL(WEIGHT) 1IMAG(WEIGHT)/(1X,F2.0,F12.4,2E14.4) )
 1600 FORMAT (1H0//)

      END
      FUNCTION SPONTY(X, PHWTS,KPHWTS, JP, NP,NPGP1, YPRED,KY)
C
C
C TITLE - SPONTY = RESPONSE, TYDE
C     GENERATES TIDE SERIES FROM PREDICTION WEIGHTS
C
C
C                    ---ABSTRACT---
C
C              SPONTY  FORMS TIME SERIES  Y(P,T) = SUM OVER H OF
C              CONJG(W(P,H))*X(P,T+H).  W(P,H)  REPRESENTS THE COMPLEX
C              WEIGHT ASSOCIATED WITH THE PTH COMPONENT OF THE REFERENCE
C              SERIES AT LEAD  H.  W(P,H)  IS OBTAINED FROM INPUT SERIES
C              PHWTS  CONSISTING OF  P,  H,  W(P,H)  MERGED (I.E. A 4
C              REAL COMPONENT SERIES).  X(P,T)  REPRESENTS THE  PTH
C              COMPONENT OF THE COMPLEX REFERENCE SERIES  X  AT TIME  T.
C              IF  KOMPLX  IS ZERO ONLY THE REAL PART OF  Y(P,T)  IS RE-
C              TAINED. THE  P  VLAUES TO BE USED ARE SPECIFIED BY ARRAY
C              JP.  UNLESS JP(1) IS ZERO, THE ONLY  P  VAULES USED ARE
C              THOSE GIVEN BY  JP(1), JP(2), ..., JP(N)  WHERE
C              N = NP(NPGP1), AND THE OUTPUT SERIES  YPRED(IPGP,T)
C              CONSISTS OF THE MERGED COMPONENT SUMS  Y(JP(I),T)
C              + Y(JP(I+1),T) + ...+ Y(JP(IP),T)  WHERE  I=NP(IPGP)+1
```

```
C              AND  IP=NP(IPGP+1) AND IPGP=1,2,...,(NPGP1-1).  IF  JP(1)
C              IS ZERO THEN ALL  P  VALUES IN  PHWTS  ARE USED AND
C              YPRED(T)  IS THE SINGLE (REAL OR COMPLEX) COMPONENT
C              SERIES CONSISTING OF THE SUM OF  Y(P,T)  OVER ALL  P.
C
C
C                          --STATISTICS--
C
C LANGUAGE   - UCSD FORTRAN 63
C EQUIPMENT  - NO SPECIFIED REQUIREMENTS
C STORAGE    - 261 WORDS FOR THE PROGRAM + 1008 WORDS COMMON (480 OF
C              WHICH ARE DUMMY) + SPACE FOR ARRAYS  X,  PHWTS,  JP,
C              NP,  YPRED (I.E. 2*KX+KPHWTS+KP+NPGP1+K12*KY WORDS, WHERE
C              K12=1 OR 2 ACCORDING AS  KOMPLX  IS ZERO OR NON-ZERO)
C SPEED      -
C AUTHOR     - MARK WIMBUSH    IGPP    AUG 1970
C LAST MOD   - MARK WIMBUSH    NOVA    APR 1972
C CATAGORIES -
C STATUS     -
C LIBRARY ROUTINES USED - NONE
C SYSTEM ROUTINES USED - IROUND, XINTF, AIMAG, FLOATF
C
C
C                          ----USAGE----
C
C SAMPLE CALL
C     J = SPONTY(X,PHWTS,KPHWTS,JP,NP,NPGP1, YPRED,KY)
C
C NOTE       - DIMENSION OF REFERENCE SERIES  X  IS
C              LIMP*INT(1.5+(EX-SX)/D) (COMPLEX)
C            - DIMENSION OF COMPONENT NUMBER SERIES  JP  IS  NP(NPGP1)
C            - DIMENSION OF PREDICTED SERIES  YPRED  IS
C              KY=INT(1.5+(EY-SY)/D)   (YPRED  CONSIDERED COMPLEX
C              UNLESS  KOMPLX=0)
C
C
C INPUTS
C
C     X(I)     COMPLEX MERGED REFERENCE SERIES ON WHICH THE PREDICTION
C              WEIGHTS ARE BASED
C
C     PHWTS(I) ARRAY CONSISTING OF  P  VALUE,  H  VALUE, COMPLEX PRE-
C              DICTION WEIGHT  W(P,H)  MERGED TOGETHER (I.E. 4 REAL
C              COMPONENTS MERGED)
C
C     KPHWTS   DIMENSION OF ARRAY  PHWTS  (SHOULD BE 4*NPH  WHERE  NPH
C              IS THE TOTAL NUMBER OF  P,H  COMBINATIONS IN  PHWTS)
C
C     JP(I)    ARRAY OF  P  VALUES (X  COMPONENT NUMBERS) TO BE USED IN
C              FORMING THE PREDICTED TIDE
C
C     NP(I)    NP(1) = 0
C              NP(I .GT. 1) IS THE NUMBER OF THE LAST TERM IN THE
C              (I-1)TH   GROUP OF  ARRAY  JP
```

```
C
C     NPGP1    DIMENSION OF ARRAY  NP
C
C ----COMMON /RESTOR/
C
C     INISHL   INISHL .NE. 0  INDICATES THAT THE CALL IS THE INITIAL
C              CALL IN THIS COMPUTATION
C              INISHL .EQ. 0  INDICATES THAT THE CALL IS NOT THE
C              INITIAL CALL
C
C     SX       START TIME OF SERIES  X
C
C     EX       END TIME OF SERIES  X
C
C     SY       START TIME OF SERIES  YPRED
C
C     EY       END TIME OF SERIES  YPRED
C
C     D        UNIFORM TIME INCREMENT OF SERIES  X  AND SERIES  YPRED
C
C     LIMP     NUMBER OF COMPLEX COMPONENTS MERGED IN SERIES  X
C
C     KOMPLX   KOMPLX .NE. 0  INDICATES THAT THE COMPLEX PREDICTED
C              SERIES IS REQUIRED
C              KOMPLX .EQ. 0  INDICATES THAT ONLY THE REAL PART OF THE
C              PREDICTED SERIES  IS TO BE RETAINED
C
C
C OUTPUTS
C
C     YPRED(I) TIME SERIES OF TIDE PREDICTIONS, HAVING  NPGP1-1 MERGED
C              (REAL OR COMPLEX) COMPONENTS IF  JP(1) .NE. 0,  OTHERWISE
C              HAVING ONE (REAL OR COMPLEX) COMPONENT
C
C     KY       DIMENSION OF SERIES  YPRED (YPRED CONSIDERED COMPLEX
C              UNLESS  KOMPLX=0)
C
C     SPONTY   =0. IF INPUT ITEMS HAVE VALID VALUES
C              =1. IF VALUES IN  NP  ARE NOT IN INCREASING ORDER
C              =2. IF THE  P@S  GIVEN IN ARRAY  JP  DO NOT MATCH THE
C              P@S  GIVEN IN SERIES  PHWTS
C              =3. X  SERIES STARTS TOO LATE TO ACCOMODATE MINIMUM LEAD
C              =4. IF  SY  IS NOT ONE OF THE TIMES OF SERIES  X
C              =5. X  SERIES ENDS TOO EARLY TO ACCOMODATE MAXIMUM LEAD
C              =6. IF  NP(1) .LT. 0
C              =7. IF THERE ARE NO  P  GROUPS, THAT IS  NPGP1 .LT. 2
C              =8. IF  KPHWTS  INVALID (.LE.0  OR NOT A MULTIPLE OF 4)
C              =9. IF  (TIME INTERVAL)/(END TIME - START TIME)  IS
C              NEGATIVE FOR SERIES  YPRED
C
C
C EXAMPLES
C
C
```

```
C PROGRAM FOLLOWS BELOW
C
      COMMON /RESTOR/ INISHL,SX,EX,SY,EY,D,LIMP,KOMPLX
      COMMON /WORKIN/ RSVP(500),JPOPHW(125),JPHW(125),JPH(125),MPH(125)
c\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
      COMPLEX X
      DIMENSION X(1),PHWTS(KPHWTS),JP(1),NP(NPGP1),YPRED(1)
C         (SEE @NOTE@ ABOVE FOR TRUE DIMENSIONS OF  X,JP,YPRED)
      EQUIVALENCE (JPIP,RSVP(1)),(JXY,RSVP(2)),(J,RSVP(3)),
     1 (INC,RSVP(4)),(LP,RSVP(5)),(MPH1,RSVP(6)),(MPH2,RSVP(7)),
     2 (NPGI,RSVP(8)),(NPGINT,RSVP(9)),(NPH,RSVP(10)),(NPHT,RSVP(11)),
     3 (NPHW,RSVP(12)),(NP1,RSVP(13)),(NP2,RSVP(14)),(DDEMSY,RSVP(15)),
     4 (DD1E3,RSVP(16)),(HMAX,RSVP(17)),(HMIN,RSVP(18)),(XY,RSVP(19))
      DATA DELTO,DELO/0.8,0.8/

      SPONTY=0.
      IF(SY.NE.EY) GO TO 10
      KY=1
      GO TO 30
   10 DDEMSY=D/(EY-SY)
      IF(DDEMSY.GT.0.) GO TO 20
      SPONTY=9.
      RETURN
   20 KY=INT(1./DDEMSY+1.5)
C         SKIP NEXT SECTION IF NOT INITIAL CALL
   30 IF(INISHL.EQ.0) GO TO 100
      IF(JP(1).NE.0) GO TO 34
      NP(1)=0
      NP(2)=1
      NPGP1=2
   34 IF(NP(1).LT.0) SPONTY=6.
      IF(NPGP1.LT.2) SPONTY=7.
      IF(MOD(KPHWTS-3,4)-1.NE.0) SPONTY=8.
      IF(SPONTY.NE.0.) RETURN
      NPHW=0
      NPH=0
      MPH(1)=0
      HMIN=PHWTS(2)
      HMAX=PHWTS(2)
      DO 40 IPHWTS=1,KPHWTS,4
        IF(PHWTS(IPHWTS+1).LT.HMIN) HMIN=PHWTS(IPHWTS+1)
        IF(PHWTS(IPHWTS+1).GT.HMAX) HMAX=PHWTS(IPHWTS+1)
        NPHW=NPHW+1
C---      (JPOPHW(I) IS THE I@TH  P  IN MERGED INPUT SERIES  PHWTS)
   40 JPOPHW(NPHW)=INT(PHWTS(IPHWTS))

C---  FOR EACH NEEDED P,H COMBINATION COMPUTE OFFSET JPH IN SERIES X

      DO 80 IPGP1=2,NPGP1
        NP1=NP(IPGP1-1)+1
        NP2=NP(IPGP1)
        IF(NP2.LT.NP1) SPONTY=1.
        DO 70 IP=NP1,NP2
          NPHT=NPH
```

```
          JPIP=JP(IP)
          DO 50 IPHW=1,NPHW
            IF(JPOPHW(IPHW).NE.JPIP.AND.JP(1).NE.0) GO TO 50
            NPH=NPH+1
C---          (JPHW(I) IS NUMBER IN THE MERGED INPUT SERIES  PHWTS
C---           OF THE I@TH SELECTED  P,H,W  GROUP)
            JPHW(NPH)=IPHW
            JPH(NPH)=JPOPHW(IPHW)+IROUND(PHWTS(4*IPHW-2)/D)*LIMP
   50     CONTINUE
          IF(NPHT.NE.NPH) GO TO 70
          SPONTY=2.
   70   CONTINUE

C---      (MPH(I+1) IS THE SELECTION NUMBER OF THE LAST  P,H,W
C---        GROUP ASSOCIATED WITH THE LAST  P  IN THE I@TH GROUP
C---        IN THE STATEMENT LIST   (MPH(1)=0)  )

        MPH(IPGP1)=NPH
   80 continue

      INC=1
      IF(KOMPLX.NE.0) INC=2
      NPGI=(NPGP1-1)*INC
  100 XY=SY-SX
      IF(-XY-HMIN.GT.DELTO) SPONTY=3.
      XY=XY/D
      JXY=IROUND(XY)
      IF(ABS(XY-FLOAT(JXY)).GT.DELO) SPONTY=4.
      IF(EY+HMAX-EX.GT.DELTO) SPONTY=5.
      IF(SPONTY.NE.0.) RETURN
      JXY=JXY*LIMP
      NPGINT=NPGI*KY
C        SET TO ZERO ARRAY YPRED

      DO 110 IT=1,NPGINT
        YPRED(IT)=0.
  110 continue

      LP=1

C---     COMPUTE PREDICTED SERIES AND STORE IN ARRAY YPRED IN MERGED
C---     FORM

      DO 140 IT=1,KY
        DO 130 IPGP1=2,NPGP1
          MPH1=MPH(IPGP1-1)+1
          MPH2=MPH(IPGP1)
          DO 120 IPH=MPH1,MPH2
            J=JXY+JPH(IPH)
            IPHW=JPHW(IPH)
            YPRED(LP)=YPRED(LP)
     1      +PHWTS(4*IPHW-1)*REAL(X(J))+PHWTS(4*IPHW)*AIMAG(X(J))
            IF(KOMPLX.NE.0) YPRED(LP+1)=YPRED(LP+1)
     1      +PHWTS(4*IPHW-1)*AIMAG(X(J))-PHWTS(4*IPHW)*REAL(X(J))
```

```
120       CONTINUE
130     LP=LP+INC
140   JXY=JXY+LIMP
      RETURN
      END
! @FOR,IS TIDE.TADM,TIDE.TADM
      FUNCTION TADM(PHW,LI, ORAY,LO, FS,DELF,CMPNT, KEEP)
      DIMENSION PHW(LI),ORAY(LO)
      DIMENSION           FREQ(49)
      REAL*8 KSYMB(50)
C
C        DARWIN SYMBOLS AND FREQUENCIES STORED IN ORDER OF ASCENDING
C        SIGNIFICANCE.
C
      DATA KSYMB /2HS8, 2HS6, 2HS4, 2HS3, 4H2SM2, 2HS1, 3HMSF, 3HSSA,
     1 5H2MSN8, 4HMSN6, 4H2SM6, 3HMK4, 3HSO3, 3HSK3, 4HMNS2, 3H2Q1,
     2 3HPI1, 6HSIGMA1, 3HRO1, 3HOO1, 2HMF, 2HSA, 6H2(MS)8, 4H2MN6,
     3 3HMN4, 4H2MK3, 3H2N2, 2HT2, 2HL2, 3HMU2, 2HM1, 2HJ1, 4H3MS8,
     4 4H2MS6, 3HMS4, 3HMK3, 3HNU2, 2HQ1, 2HM8, 2HM6, 2HM4, 2HM3, 2HK2,
     5 2HN2, 2HP1, 2HO1, 2HS2, 2HK1, 2HM2, 1H /
C
      DATA FREQ / 8.0, 6.0, 4.0, 3.0, 2.067726385, 1.0, .0677263854,
     1 .0054758186, 7.760529198, 5.828255583, 5.932273615, 3.937749433,
     2 2.929535705, 3.002737909, 1.828255583, .8569524129, .9945243121,
     3 .8618093199, .8981009661, 1.075940113, .073202204, .0027379093,
     4 7.864547229, 5.760529198, 3.828255583, 2.86180932, 1.859690322,
     5 1.997262221, 1.968565261, 1.864547229, .9664462631, 1.039029556,
     6 7.796820844, 5.864547229, 3.932273615, 2.935011524, 1.900838875,
     7 .8932440591, 7.729094458, 5.796820844, 3.864547229, 2.898410422,
     8 2.005475819, 1.895981968, .9972620907, .9295357053, 2.0,
     9 1.002737909, 1.932273615/
C
 1    FORMAT(F6.4)
 2    FORMAT(1H0,F10.3,F10.4,F11.6,F13.6,F12.6,F13.6,F10.3,2(5X,A7,
     1 8X))
 3    FORMAT(1H0,8X,17HF R E Q U E N C Y,9X,1H*,12X,19HA D M I T T A N C
     1 E,14X,1H*,8X,21HT I D A L   L I N E S/
     2 44X,28H$NAMOBS LEADS $NAMREF BY DEG/
     3 6X,37HCPY       CPM        CPD      *    REAL,8X,67HIMAG    *  AMPLI
     4TUDE     DEG   *  MOST SIGNIFICANT  *   MOST CENTRAL/
     5 35X,1H*,22X,1H*,22X,1H*,13X,8HCPD    *,12X,3HCPD)
 4    FORMAT(1H0)
C
C        LO IS LENGTH OF ORAY
C        LI IS LENGTH OF PHW
C        THE LENGTH OF PHW SHOULD BE A MULTIPLE OF 4
      TADM = 1.
      IF(MOD(LI,4) .NE. 0) RETURN
C
      PRINT 3
      PI = -3.1415926536
      F = FS
```

```
C
      DO 3000 N1=1,LO,2
      N2=N1 + 1
      OREAL = 0.
      OIMAG = 0.
C
      DO 2000 K1=1,LI,4
      K2=K1 + 1
      K3 = K2 + 1
      K4 = K3 + 1
      IF(PHW(K1) .NE. CMPNT) GO TO 2000
      ALPH = PHW(K2) * PI * F / 12.
      CF = COS(ALPH)
      SF = SIN(ALPH)
      OREAL = OREAL + PHW(K3) * CF + PHW(K4) * SF
      OIMAG = OIMAG + PHW(K4) * CF - PHW(K3) * SF
 2000 CONTINUE
C
C        CPM = 27.321582          TROPICAL MONTH
      CPM = F * 27.321582
C        CPY = 365.25             JULIAN YEAR
      CPY = F * 365.25
C      NOTE THAT A TROPICAL YEAR IS 365.24219879 FOR THE YEAR 1900 AND
C        SHOULD HAVE .00000600 SUBTRACTED FOR EACH CENTURY AFTER 1900.
      R = SQRT(OREAL**2 + OIMAG**2)
      PSI = ATAN2(OIMAG,OREAL) * 57.29578
      FMAX = F + DELF/2.
      FMIN = FMAX - DELF
C
C        PICK THE DARWIN SYMBOL AND FREQUENCY OF BOTH THE MOST CENTRAL
C        LINE AND THE MOST SIGNIFICANT LINE CONTAINED IN ANY BAND
C        F PLUS OR MINUS DELF.  SIGNIFICANCE IS TAKEN TO BE THAT IMPLIED
C        BY THE ORDERING OF THE LINES IN THE HARMONIC CONSTANTS TABLES
C        OF THE INTERNATIONAL HYDROGRAPHIC BUREAU (MONACO), WITH
C        SPECIES INTERSPERSED.
C
C        SET  DIFF = LARGE NUMBER
      DIFF =10.**35
      JNEAR = 50
      JSIG=50
C
      DO 2200 J=1,49
      IF(FREQ(J) .GT. FMAX) GO TO 2200
      IF(FREQ(J) .LT. FMIN) GO TO 2200
      JSIG = J
      T = ABS(F - FREQ(J))
      IF(T .GT. DIFF) GO TO 2200
      DIFF = T
      JNEAR = J
 2200 CONTINUE
C
      IF(JNEAR .EQ. 50) GO TO 3400
      PRINT 22, CPY,CPM,F,OREAL,OIMAG,R,PSI,KSYMB(JSIG),FREQ(JSIG),
     1 KSYMB(JNEAR),FREQ(JNEAR)
```

```
   22 FORMAT(1H0,F10.3,F10.4,F11.6,F13.6,F12.6,F13.6,F10.3,
     1 2(5X,A7,F6.4,2X))
 2400 IF(KEEP .EQ. 0) GO TO 2500
      ORAY(N1) = OREAL
      ORAY(N2) = OIMAG
 2500 F = F + DELF
 3000 CONTINUE
C
      TADM = 0.
      PRINT 4
      RETURN
 3400 PRINT 2,CPY,CPM,F,OREAL  ,OIMAG   ,R,PSI,KSYMB(JSIG),KSYMB(JNEAR)
      GO TO 2400
      END
       SUBROUTINE HG(ORAY,LO,nCONST,NNC,MORDER,NTYPE)
C
C       TITLE - HG
C       COMPUTES LINE AMPLITUDES IN CENTIMETERS AND GREENWICH
C       PHASE FROM RESPONSE ADMITTANCES
C           AUTHOR- MARK WIMBUSH    NOVA 1975
C
      DIMENSION HPOT02(2),HPOT(2,2,2),HPOTX3(2,3,2),ORAY(LO),
     &          NCONST(NNC)
      DATA HPOT02/3.1,6.663/
      DATA HPOT/26.221,36.878,5.02,12.203,63.192,7.996,12.099,29.4/
      DATA HPOTX3/1.3871,0.05969,0.2314,0.22144,0.55741,
     &             0.048,0.399,0.146,0.389,0.359,0.210,0.765/

      print 120

      DO N = 1,NNC
        NC = NCONST(N)
        OREAL = ORAY(2*NC-1)
        OIMAG = ORAY(2*NC)
        R = SQRT(OREAL**2+OIMAG**2)
        PSI = ATAN2(OIMAG,OREAL)*57.29578
        G = AMOD(180.0*FLOAT(MORDER)-PSI,360.0)

        IF (NTYPE.EQ.10) then
          J = 1
          IF (NCONST(1).EQ.1) J = 2
          H = R*HPOT(N,J,MORDER)
        else IF (NTYPE.EQ.-10) then
          H = R*HPOT02(N)
        else
          IF (NTYPE.GT.0) then
            I = 2
            J = NTYPE
          else
            I = 1
            J = 1-NTYPE
          endif
          H = R*HPOTX3(N,J,I)
        endif
```

```
        print 160,NC,H,G
      enddo


      RETURN

120   FORMAT('0',18X,'H',20X,'G')
160   FORMAT('0',1X,I3,F20.5,F20.3)

      END
```

## 3.9 FILTER_NAMES.M

```
%*******************************************************
% filter_names.m
%
% filters the ies records specified in the file names.m
% 24-Jan-1990
%*******************************************************
% enter the file names, bints and filter coefficents from quasi-control
% file names.m.
%
% z     = array of seacor file prefixes
% bints = vector containing B-intercepts
% b,a   = butterworth filter coefficents, as in signal processing tool box
%                              documentation, or 'help butter' in Matlab
%
echo off
names                               % get z,bints,b, and a
for i=1:length(bints)               % loop through all files

eval(['load ',z(i,:),'.seacor'])    % load file.  Eval(t) is a text macro
                                    % faciltiy.  It causes text string in
                                    % t to be interpreted as a matlab
                                    % command or expression. 'help eval'
                                    % or 3-39 in the manual.

eval(['y = ',z(i,:),'(:,2);'])      % assign travel times to vector y
eval(['t = ',z(i,:),'(:,4);'])      % assign time to vector t

k = rampf(y);                       % remove ramp (line from first to last
y = y - k;                          %               point) not a Matlab m-file

y = filter(b,a,y);                  % filter forward. See 3-47 in Matlab
                                    % manual or 'help filter'

y = flipx(filter(b,a,flipx(y)));    % filter backwards.  flipx flips a
                                    % matrix about the x axis. here it
                      % is used to reverses the order of a
                                    % column vector (and will do nothing to a
                                    % row vector).  also see flipy.

y = y + k;                          % return ramp


y = y(41:length(y)-40);             % remove regions with possible
t = t(41:length(t)-40);             % transient ringing

[sy,st] = subsample(y,t);           % subsample at even 6 hourly
                                    % increments. not a Matlab m-file

sy=-19800*sy + bints(i)*ones(length(sy),1);
                                    % calibrate to Z_12 depth. ones(n,m)
                                    % creates a nxm matrix of ones. 3-89

```

```
q = [st,sy];                          % save to mat file
eval(['save ',z(i,:),'.z12star q /ascii'])
eval(['clear 'z(i,:)])
end                                   % end for loop
```

## REFERENCES

Bialek, E.(compiled by) 1966. Handbook of Oceanographic Tables. U.S. Naval Oceanographic Office, Special Publication-68. Washington, D.C.

Chaplin, G. and D. R. Watts. 1984. Inverted echo sounder development. *IEEE Oceans '84 Proceedings. 1*, 249–253.

Munk, W. H. and D. E. Cartwright. 1977. Tidal spectroscopy and prediction. *Phil. Trans. Roy. Soc. London, 259*, 533-581.

Press, W.H., B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. 1988. *Numerical Recipes* Cambridge University Press, New York.

Rossby, T. 1969. On monitoring depth variations of the main thermocline acoustically. *J. Geophys. Res. 74*, 5542-5546.

Schureman, P. 1941. *Manual of Tide Observations.* U.S. Department of Commerce. Coast and Geodetic Survey, Washington, D.C. 92p.

Watts, D. R. and W. E. Johns. 1982. Gulf Stream meanders: observations on propagation and growth. *J. Geophys. Res. 87*, 9467-9476.

Watts, D. R. and H. Kontoyiannis. 1990. Deep-ocean bottom pressure measurements: Drift removal and performance, *J. Atmos. Ocean. Technol.*, 7, 296-306.

Watts, D. R. and H. T. Rossby. 1977. Measuring dynamic heights with inverted echo sounders: Results from MODE. *J. Phys. Oceanogr. 7*, 345-358.

# A COMMAND PROCEDURES

Most of the IES processing programs are not interactive, and a VMS command procedure is usually used to associate the programs logical units with the proper input and output files.

An example command procedure is listed below. This procedure assigns all the necessary files to logical units and runs MEMOD. By defining FOR005 to be [.ctrl]'p1'.ctrl the logical unit 5 is then defined to be the control file (here p1 is a parameter passed to the procedure at the command line). For example, @MEMOD90 SN078 IES90H1 calls the procedure below and passes 'SN078' and 'IES90H1' as the parameters P1 and P2.

```
$!
$!                           MEMOD90.COM
$! Command file for running the memod program on a specific data set
$!
$!  INPUT/OUTPUT UNITS USED:
$!  KR      (UNIT   5) - CONTROL INPUT
$!  KW      (UNIT   6) - LOG OUTPUT
$!  KWDA    (UNIT   7) - TT1 MODE/MEDIAN DISK OUTPUT DATASET
$!  KWDB    (UNIT   8) - TT2 MODE/MEDIAN DISK OUTPUT DATASET
$!  KWLA    (UNIT   9) - TT1 LISTING OF STATISTICS
$!  KWLB    (UNIT  10) - TT2 LISTING OF STATISTICS
$!  KRBUNS (UNIT  11) - INTEGER INPUT OF BUNS DATA
$!
$ DEFINE FOR005 [.ctrl]'p1'.CTRL
$ DEFINE FOR006 'p2'.MEMOD_LOG
$ DEFINE FOR007 'p2'_TT1.mode
$ DEFINE FOR008 'p2'_TT2.Mode
$ DEFINE FOR009 'p2'_TT1.LIST
$ DEFINE FOR010 'p2'_TT2.LIST
$ DEFINE FOR011 [cruise.data.ies]'p1'.BUNS
$ RUN MEMOD_jul89
```

# B MASS PRODUCTION

The time required to process large numbers of IESs can be shortened by writing code which creates the control file from the minimum of information necessary. Much of the information that is in the control files is the same for all the instruments in a deployment; this information can be set once within DATA statements of this 'driver' program, while the variable information can be condensed to a single line of a 'list' file. The programs PUNS, FILL, DETIDE, and SEACOR are run by 'drivers' in this fashion.

Below is an example of an input list file for a SEACOR driver program (which is also listed below). A row of stars separates the header from the data. All lines above the row of stars are ignored. For each line read below the stars, the program writes a control file and then calls a command procedure that runs SEACOR.

```
list for seacor_bash

instrument, site, type(p for pressure), number of records, number
of years spanned for the time series, leap year?(YE or NO), ditto, cruise
number

  examples from OC207:
66 I2 P 17015 2 YE NO 207
37 J2   17019 2 YE NO 207

starting en216
F3 and B2 were lost
*************************************************
57 A2   18587 2 NO NO 216
37 F1   21169 2 NO NO 216
62 F2   17798 2 NO NO 216
37 G1   21615 2 NO NO 216
67 G2 P 21135 2 NO NO 216
55 G3 P 21462 2 NO NO 216
76 G4   20054 2 NO NO 216
44 H1   21077 2 NO NO 216
71 H2 P 17316 2 NO NO 216
53 H3 P 17526 2 NO NO 216
```

It is simple to process one or many files. In this example 10 files are processed. After processing these ten, a new IES record can be processed by relocating the row of stars to the last line and adding the new IES information beneath it. All the previously processed lines are incorporated into the header and consequently excluded.

The program that uses the file above is listed here as an example.

```
C***
C*** New program used to generate control files for and run seacor
C*** name    = a string used to submit the command procedure call to DCL
C*** filename= string used to generate control file names
C*** file    = string usedto generate the file suffix ie. ies90b5_216. This
```

```
C***            string is used in name.
C*** headr   = used for the header card of the control file
C*** ...
C***
character*60 name,filename,file
character*60 headr
character*3 cruise
character*2 site,instr,yesno(2)
character*1 type
integer recs,yearspan,status,lib$spawn,size,str$trim
logical exist

C****
C**** Get the file name (name) of file containing all the pertinent
C**** information to make the control file.  This consists of:
C****    The instrument serial number, site, type of instrument (pressure
C****    or not), the total number of records, the length of the record,
C****    the number of years spanned by the record, whether or no the year
C****    was a leap one, ditto, the cruise number
C****
C****    fortran expressed as instr, site, type, recs, yearspan, yesno(2),
C****                         cruise.
C****
C****
C**** The file has a header. A row of stars indicates that the next line is
C****   data containing the above entries.
C****
C****

2 write(5,*) 'input file name:'
read(5,fmt='(Q,A)') name_size, name(1:name_size)
inquire(file=name(1:name_size), exist=exist)

if (exist) then
open(unit=19,file=name(1:name_size),form='formatted',status='old')
else
type*, 'file not found.'
goto 2
endif

1 read(19,fmt='(Q,A)') name_size,name(1:name_size)
if (name(1:1).ne.'*') goto 1

3 read(19,100,end=200)
     /  instr,site,type,recs,yearspan,yesno,cruise
100 format(a2,x,a2,x,a1,x,i5,x,i1,x,a2,x,a2,x,a3)

C******
C****** create file name, open it, and write appropriate cards for
C****** the given control file flag.
C******
filename='[cruise.seacor.ctrl]'
file='ies90'
fsize=5
```

```
status = str$trim (filename,filename,size)
if (.not.status) call lib$signal(%val(status))
if ((type.ne.' ').or.(type.eq.' ')) then
file=type//'_ies90'
fsize=6
filename=filename(1:size)//type
size=size+1
endif
c filename = filename(1:size)//'ies90'//site//'_'//cruise//'.ctrl'
filename = filename(1:size)//'ies90'//site//'.ctrl'
c file=file(1:fsize)//site//'_'//cruise
file=file(1:fsize)//site
open(unit=20,file=filename,form='formatted',status='new')
headr=''''//site//'  SNO'// instr//'  '//type//'  '//'
     /    EN'//cruise//''''
write(20,103) headr,recs,yearspan,yesno(1),yesno(2)
if (((ichar(site(1:1)).ge.97).and.(ichar(site(1:1)).le.99)).or.
     /  ((ichar(site(1:1)).ge.65).and.(ichar(site(1:1)).le.67))) then
write(20,201) 1
else
write(20,201) 3
endif

201 format(x,'$CARD3 region= ',i1 ,' $end')

202 close(unit=20)

C****
C****  Here name is used to generate a string that passes a command
C****  procedure call and the neccessary parameter to the LIB$SPAWN
C****  routine.  LIB$SPAWN allows the execution of the DCL procedure
C****  from within this program.
C****
C****  go_seacor simply executes seacor89.com and imprints the log file
C****
c name='@go_seacor '//file
name='@seacor90 '//file
status=lib$spawn(name)
C****
C**** get next instrument
C****
goto 3
200 close(unit=19)

103 format(' $CARD1'/'  HEADR='a60/' $END'/' $CARD2
     / NPTS=',i6,', NOYRS=',i1,', FRSTYR= '' ',a2,' '', SCNDYR
     /  ='' ',a2,' '' $END')
end
```

SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>Unclassified | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Distribution for public release;<br>Distribution is unlimited. |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>University of Rhode Island<br>Graduate School of Oceanography | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>Univ. of Rhode Island<br>Grad. School of Ocean.<br>6b. OFFICE SYMBOL (If applicable)<br>1122 PO | 7a. NAME OF MONITORING ORGANIZATION |
| 6c. ADDRESS (City, State, and ZIP Code)<br>South Ferry Road<br>Narragansett, RI 02882 | 7b. ADDRESS (City, State, and ZIP Code) |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION<br>Office of Naval Research<br>National Science Foundation<br>8b. OFFICE SYMBOL (If applicable) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>N00014-90-J-1568, N00014-90-J-1548,<br>OCE87-17144-90 |
| 8c. ADDRESS (City, State, and ZIP Code)<br>800 N.Quincy St., Arlington, VA 22217<br>1800 G.Street, NW, Washington, DC 20550 | 10. SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO. \| PROJECT NO. \| TASK NO. \| WORK UNIT ACCESSION NO. |

11. TITLE (Include Security Classification)
Inverted Echo Sounder Data Processing Report

12. PERSONAL AUTHOR(S)
Erik Fields, Karen Tracey and D. Randolph Watts

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Year, Month, Day) | 15. PAGE COUNT |
|---|---|---|---|
| summary | FROM 1987 TO 1991 | May 1991 | 151 |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | Inverted Echo Sounder (IES) |
| | | | |
| | | | |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)


The Inverted Echo Sounder (IES) is an instrument that acoustically monitors the depth of the main thermocline from a moored position one meter above the ocean floor. Additionally, the IESs can be equipped to measure both pressure and temperature. The standard steps for processing IES data are documented here. The effect and purpose of each step are discussed followed by a description of how to apply the computer programs that constitute the step. The FORTRAN and MATLAB codes are also supplied.


| | |
|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT<br>☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS | 21. ABSTRACT SECURITY CLASSIFICATION |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>D. Randolph Watts | 22b. TELEPHONE (Include Area Code) \| 22c. OFFICE SYMBOL |

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE