

2

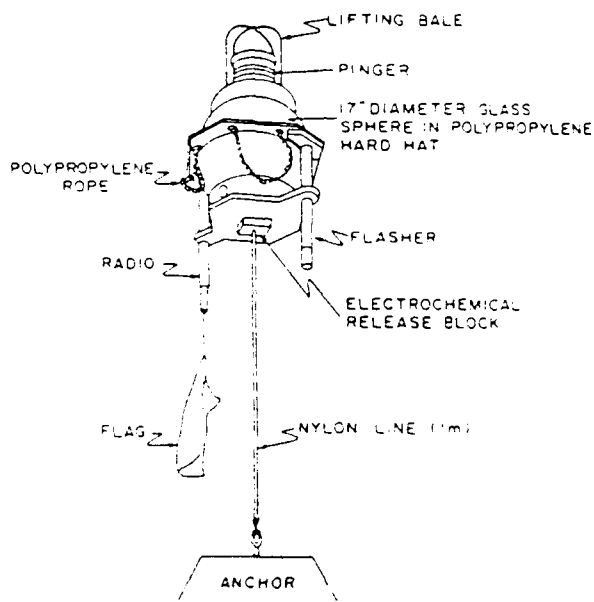
STATE SCHOOL OF OCEANOGRAPHY
UNIVERSITY OF RHODE ISLAND
NARRAGANSETT, RHODE ISLAND

DTIC
ELECTE
JUL 01 1991
S C D

Inverted Echo Sounder
Data Processing Report

GSO Technical Report No. 91-3

AD-A237 576



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 N00014-90J-1568 and N00014-90J-1548.

91-03230



91 0 24 088

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.

Available For	
DTIC	
Unannounced	
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or
	Special
A-1	

Contents

Abstract	i
List of Figures	iii
List of Tables	iii
1 Overview of IES Processing	1
2 PROGRAM DESCRIPTIONS	7
2.1 SDR	7
2.2 BUNS_AUG89.FOR and BUNS_ENGIN_AUG89.FOR	10
2.3 PUNS_MAY88.FOR	13
2.4 MEMOD_JUL89.FOR	16
2.4.1 PROCESSING OF TRAVEL TIME	16
2.4.2 PROCESSING OF ADDITIONAL SENSORS	17
2.4.3 TIME BASE	18
2.4.4 OUTPUT DATA SET	19
2.5 FILL_JAN91.FOR	24
2.6 DETIDE_AUG90.FOR	26
2.7 DESPIKE_AUG90.FOR and DESPIKE_TP.FOR	31
2.8 SEACOR_AUG90.FOR	33
2.9 RESPO_JUL88.FOR	36
2.10 FILTER_NAMES.M	40
3 PROGRAM CODE	43
3.1 BUNS_AUG89.FOR	43
3.2 PUNS_MAY88.FOR	58
3.3 FILL_JAN91.FOR	67
3.4 MEMOD_JUL89.FOR	77
3.5 DETIDE_AUG90.FOR	98
3.6 DESPIKE_AUG90.FOR	104
3.7 SEACOR_AUG90.FOR	109
3.8 RESPO_JUL88.FOR	115
3.9 FILTER_NAMES.M	142
References	145
A COMMAND PROCEDURES	147

List of Figures

1	Major Processing Steps - Travel Time	1
2	IES Data Processing Flowchart	2
3	Major Processing Steps - Pressure	5
4	Dedrifting Residual Pressure	6
5	The Detiding of Travel Time	26
6	Regions for SEACOR Correction Factors	33
7	SEACOR Correction Factors	34
8	The Effect of RESPO in the Time Domain	36
9	The Effect of RESPO in the Frequency Domain	37
10	The spherical harmonics associated with C_2^1 and C_2^2	38
11	Filter Response	40

List of Tables

1	SDR Word Format	7
2	BUNS control files	12
3	MEMOD control files	23
4	Node Factors (f)	28
5	Equilibrium Arguments ($V_0 + u$)	29

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

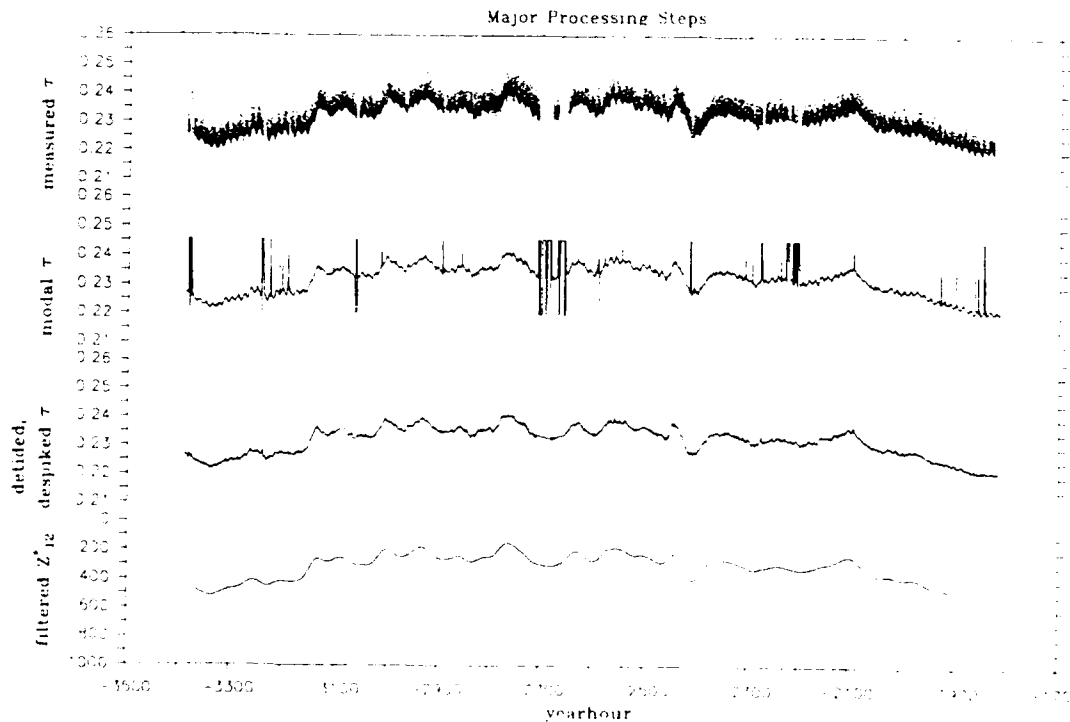


Figure 1: An IES subrecord is plotted at several processing stages to illustrate some of the major steps. The upper panel is the time series of the individual τ 's in seconds. Storms often result in periods of high scatter or a lack of returning echoes. The second plot shows the time series after a single representative τ is found for each burst of τ '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 τ calibrated to Z_{12} (in meters).

visibly noticeable steps for travel time (τ) 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.

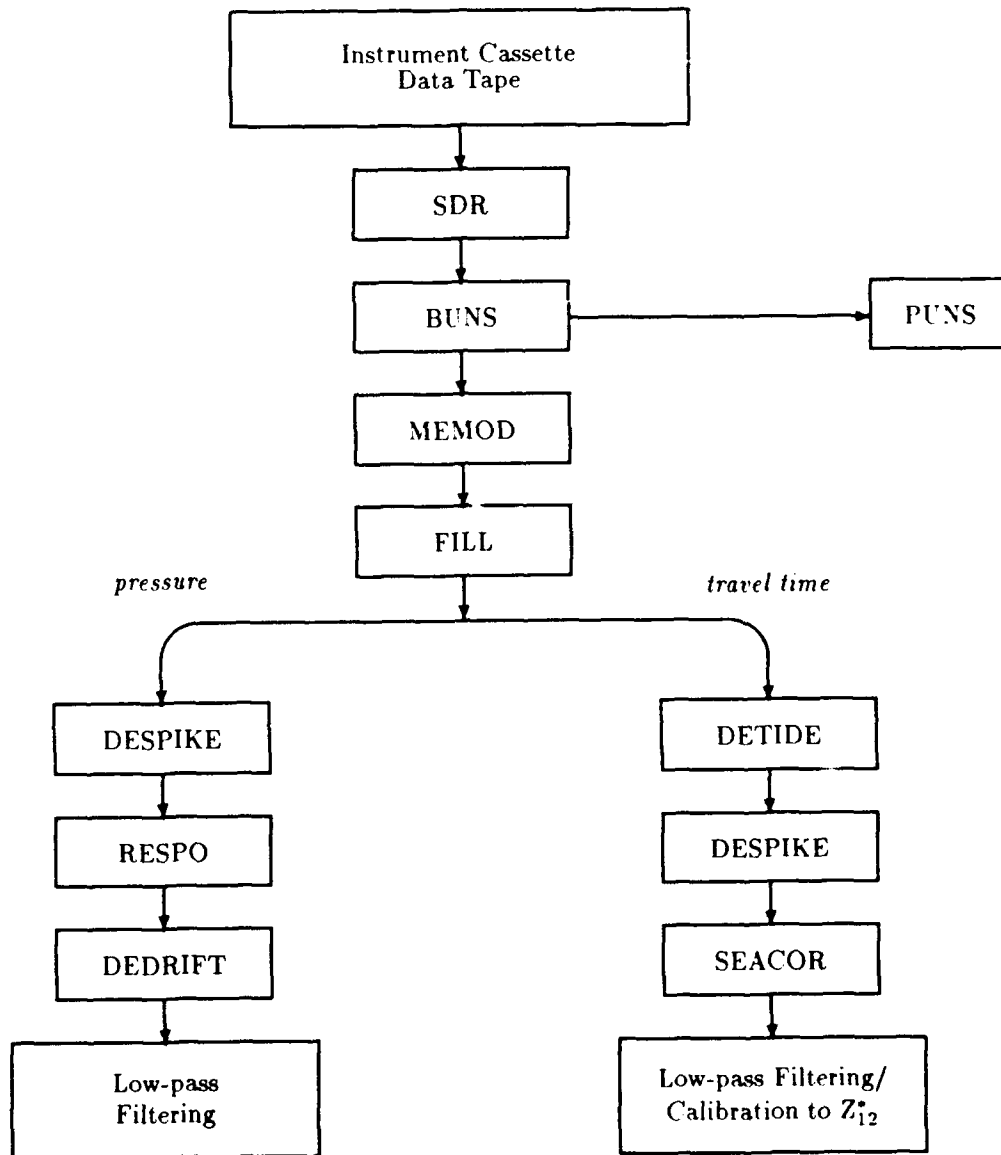


Figure 2: IES Data Processing Flowchart

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 τ is estimated from the burst of τ 's. Travel time, pressure and temperature counts are converted to units of seconds, decibars, and $^{\circ}\text{C}$ respectively.

FILL : Proper incrementation of the time base is enforced here. Missing samples are inserted using interpolated values. For PIESSs, 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 τ (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}^* , τ) 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 τ 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, $O(0.1 \text{ dbar yr}^{-1})$, 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.

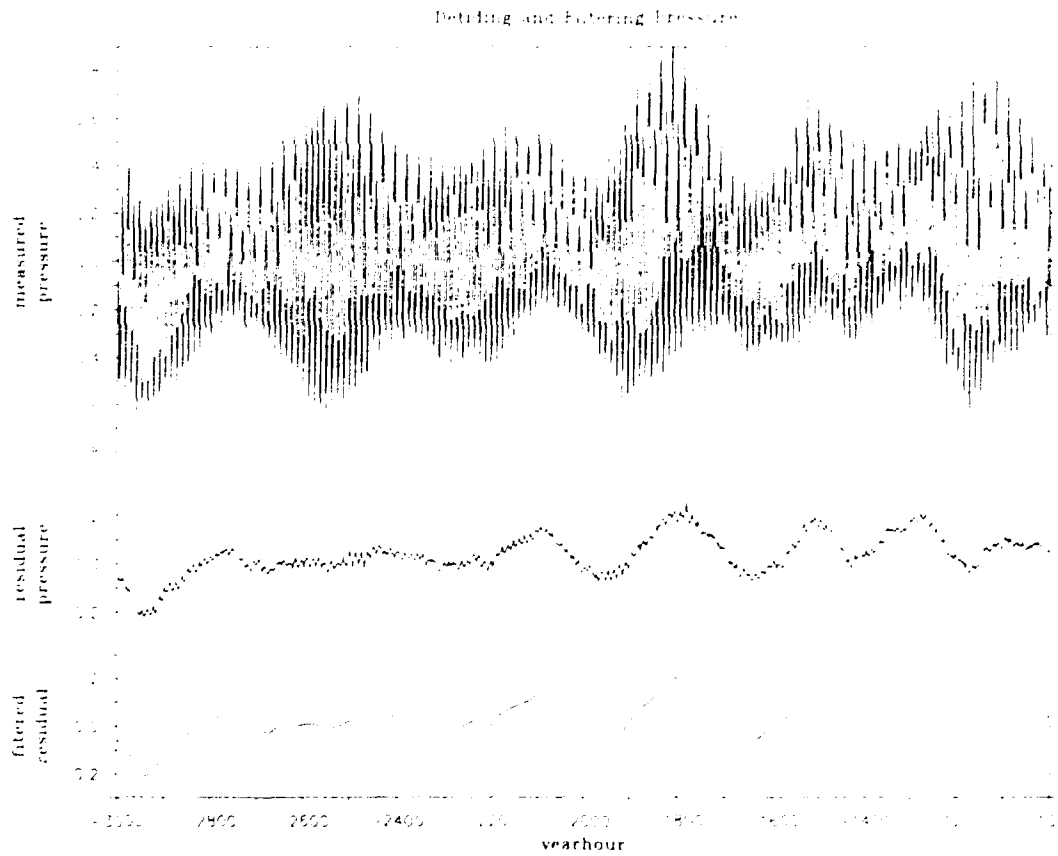


Figure 3: The detiding and filtering of a pressure are illustrated above. Pressure is offset by its mean for the entire record of 4552 dbars.

The predicted tides are then removed from the pressure records.

The pressure records are dedrifted in the manner developed by Watts and Kontoyiannis (1999) 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 form¹,

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

A design matrix for the nonlinear least-squares fit would be composed of $(e^{-\lambda t}, 1)$. 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, λ . Minimization is accomplished using the method of parabolic extrapolation and golden sections (Press et al. 1988) to optimally search for λ 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.

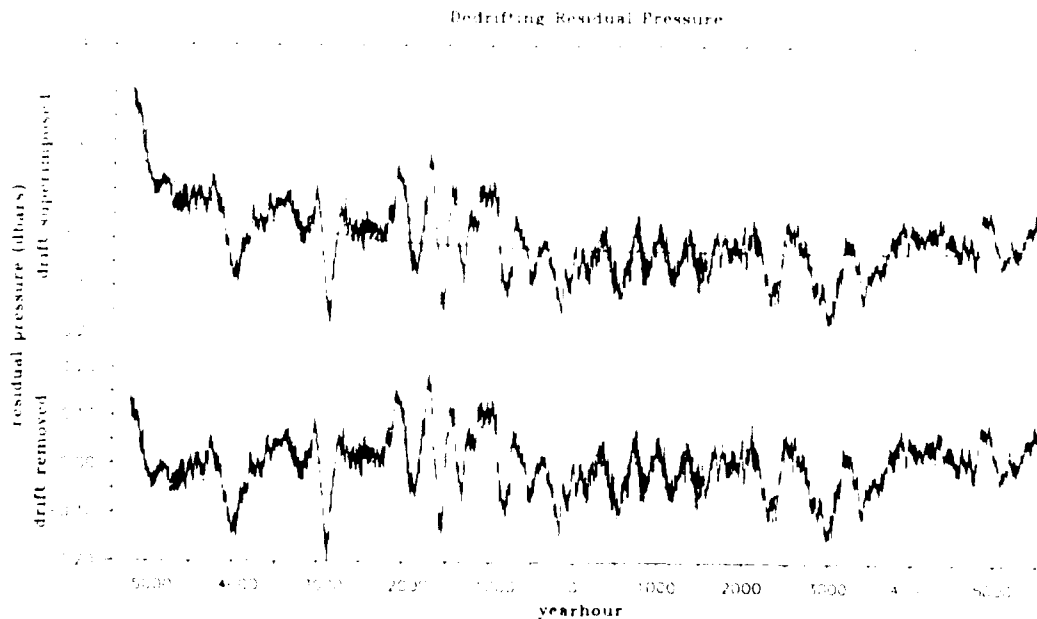


Figure 4: The residual pressure before and after the subtraction of the estimated drift (dashed).

¹When 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-NO DATA, 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/C 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 (τ) bursts within a specified range of sampling periods. This program was developed to give the user a first look at the distribution of the τ 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' τ 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 τ 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 τ 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 τ 's from each detector. The user must specify the range of τ 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 τ 's measured during a single sampling period.

LBFST - (INTEGER*4) Word number associated with the first τ 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 τ 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 τ echo detector used to convert τ from integer counts to time in decimal seconds.

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

CARD6

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

UBNDA - (INTEGER*4) Upper limit of the histogram of counts for the first τ 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 τ 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 τ counts

'DE' - decimal listing of the τ '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 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 τ 's. Then the subroutine TTMODE calculates the modal τ based on the assumption that the τ 's are members of a Rayleigh-distributed statistical population. Alternatively, the user may specify that the median τ 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 τ 's are recalculated by MEMOD by adding the appropriate power of two number of counts prior to windowing the τ 's.

The user specifies upper and lower window limits in CARD7 of the control file. If all the τ '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' τ is set equal to the limit exceeded. If the quartile range of the burst is too large, the 25th percentile τ (based on empirical evidence) is used instead of the median or

modal τ . The range, τ , and number of τ '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:

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

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

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 PIEs (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 τ 's measured during a single sampling period.

LBFST - (INTEGER*4) Word number associated with the first τ 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 τ 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 τ 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 τ counts for the first echo detector. Counts lower than LBND1 are excluded from further processing.

UBND1 - (INTEGER*4) The upper bound on the τ 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' - τ 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
IHOURL - (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 preceding 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 τ 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 τ will be determined as the median value from the subset of all measurements with counts between 7280 and 7700. This median τ 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 τ measurements for each sampling period. These measurements are stored in words 2-65 of the data record. For the first echo detector, the τ 's within the limits 99650 and 100325 will be used to determine a single τ by the mode method. The τ 's from the second detector that pass through the 1545-1840 window will be used calculate the τ by the median method. In both cases, the calculated τ 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  WTT=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, DGRPFR=2.00D+00 $END
$CARD8  IOPT=' TT1', 'MED1',2* ' ' $END
$CARD9  IYR=85, MINTH=01, IDAY=17, IHOOR=00, IMIN=59, ISEC=27, ISEQO=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  WTT=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, DGRPFR=4.00D+00 $END
$CARD8  IOPT=' TT1', 'MOD1', ' TT2', 'MED1' $END
$CARD9  IYR=82, IMINTH=07, IDAY=16, IHOOR=11, IMIN=45, ISEC=00, ISEQO=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 τ 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

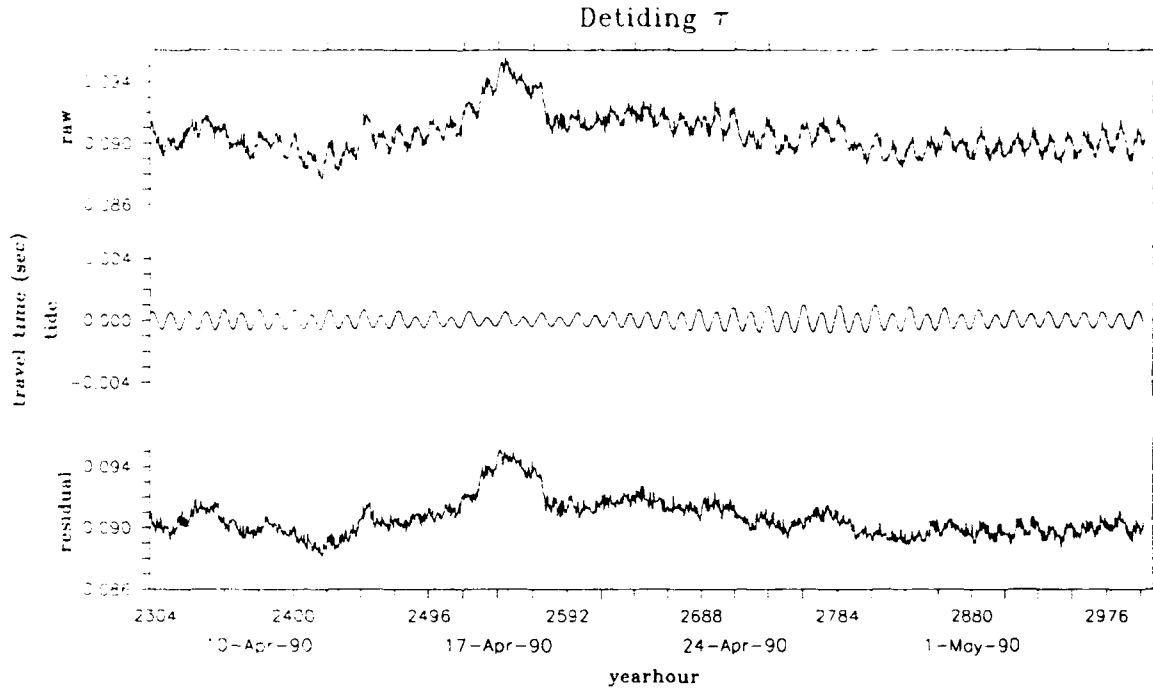


Figure 5: The measured travel time, the prediction of the tide's effect on travel time, and the 'detided' travel time.

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

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

here c_s is the speed of sound for the region of mean depth H ; δc_s is the variation in sound speed resulting from the tidal height, η . 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

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

The user supplies c_s and γ (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 ($\frac{\partial g}{\partial p} \approx 1.1 \times 10^{-5} s^{-1}$). The scaled tidal heights are subtracted from the measured τ 's to create a set of 'detided' τ 's.

The tide (η) 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 file².

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 ω_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 ($V_0 + u$) 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 τ (in seconds), detided τ (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.

²The 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 ($V_0 + u$) for the Greenwich Meridian at the beginning of each calendar year, 1973 to 1999 (Schureman, 1941)

Year	Conituent							
	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 ($V_0 + u$) 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 preceding few measurements, more rapidly than a specified rate. Figure 1 illustrates a travel time record, both before and after being processed by despiking.

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 τ . The measured τ 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).

RNAV1 - (REAL*4) Initial value for the running average of LAVG1 samples. If RNAV1 = 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

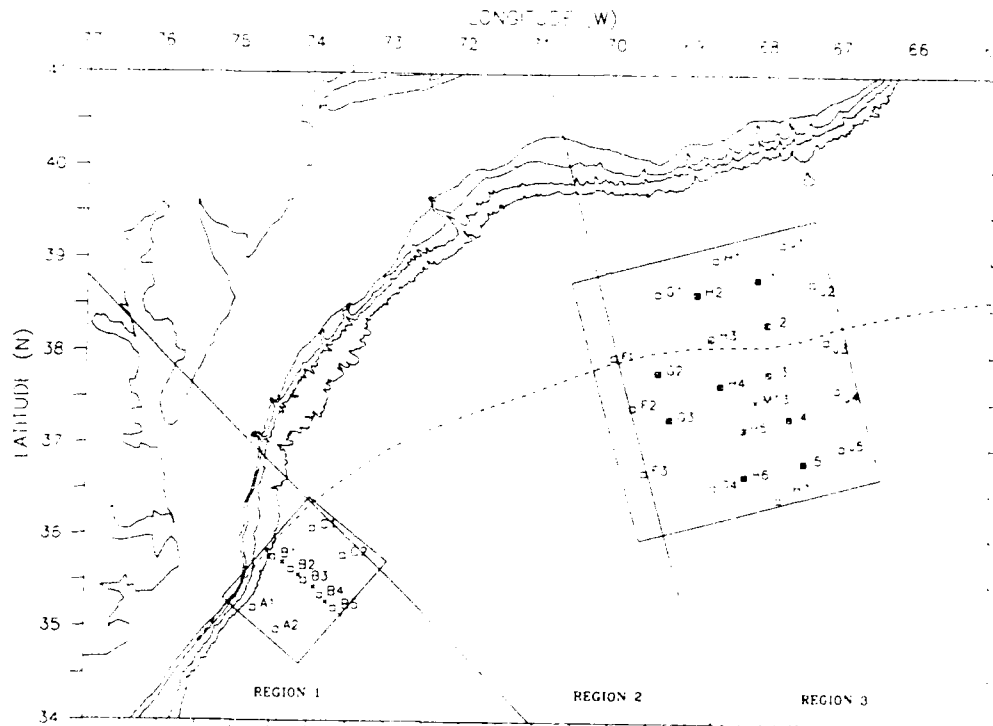


Figure 6: The three regions represent the dominant spatial variation of the seasonal corrections to travel-time.

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.

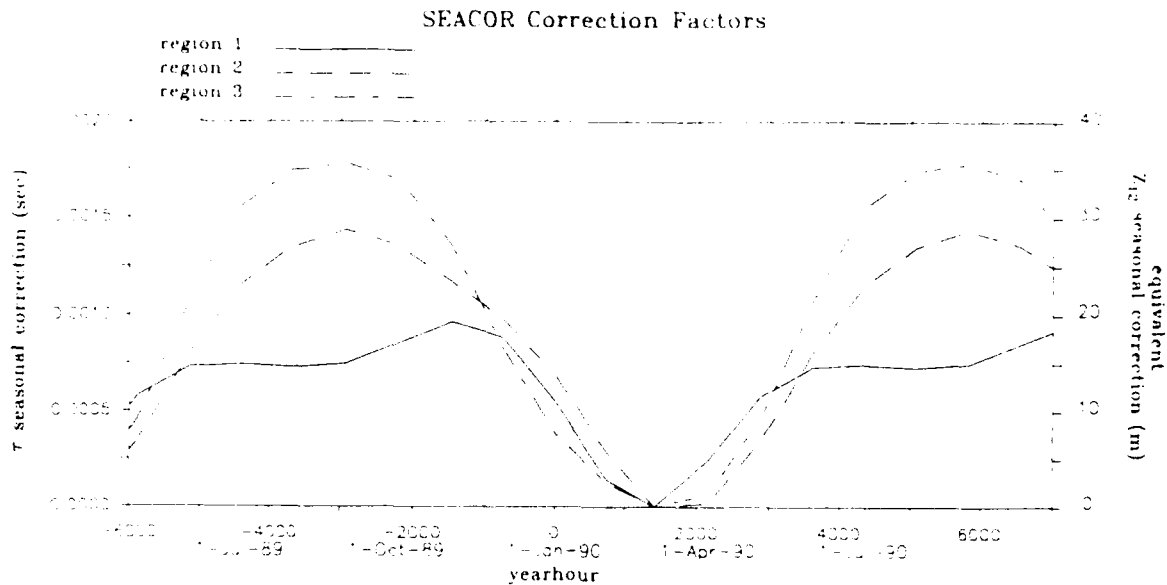


Figure 7: The seasonal correction factors for the three regions displayed in Figure 6.

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 τ '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 τ '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.

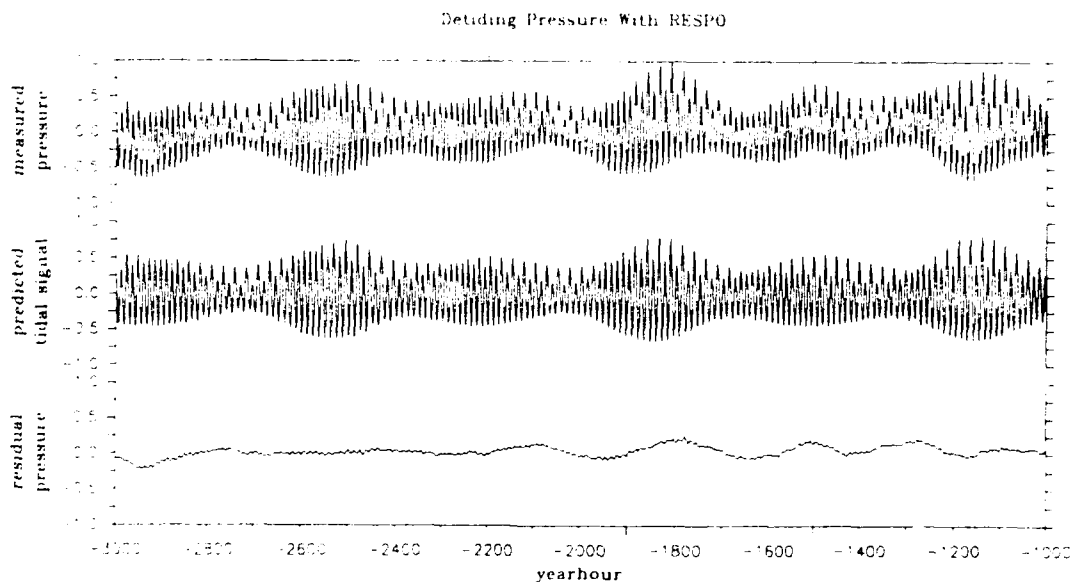


Figure 8: The uppermost panel displays the measured bottom pressure; the middle panel represents the portion of the measured bottom pressure resulting from the tide; and the bottom panel is the residual bottom pressure.

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,

$$y(t) = \sum_k \ell(\tau_k) x(t - \tau_k), \quad (3)$$

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

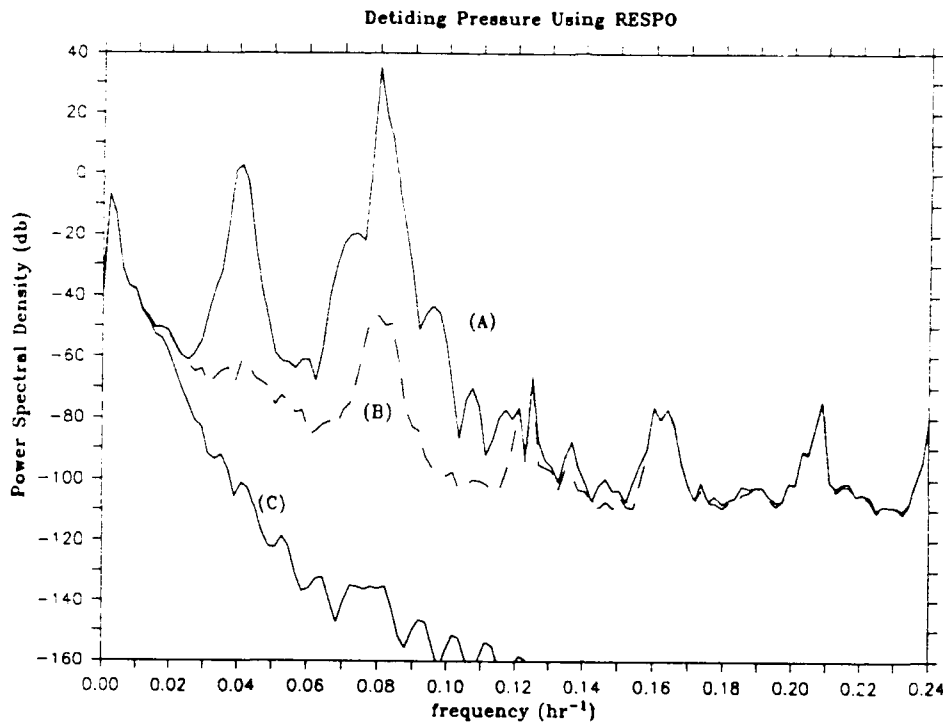


Figure 9: The power spectral density of the (A) measured bottom pressure, (B) the residual bottom pressure, and (C) the 40 hr low-pass filtered residual.

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:

$$y(t) = \sum_i \sum_k \ell_i(\tau_k) x_i(t - \tau_k), \quad (4)$$

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.

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

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_n^m , which replace ℓ , are the complex weights associated with each $C_n^m(t)$. Using the data to be detided, $(\eta(t_j))$, the weights (w_n^m) 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 ρ 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)

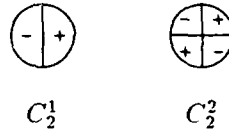


Figure 10: The spherical harmonics associated with C_2^1 and C_2^2 are viewed from the axis of rotation. Plus and minus signs correspond to bulging and flattening relative to the geoid.

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

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

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

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

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

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

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 -(CHARACTER*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 (τ) is scaled to Z_{12} .

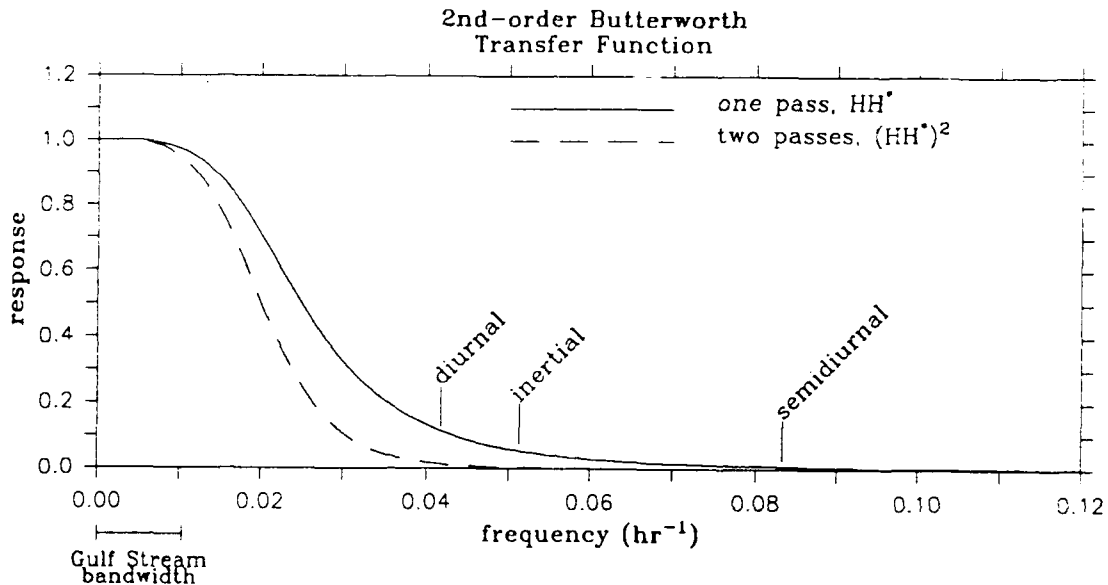


Figure 11: The transfer function for the 2nd-order Butterworth filter is illustrated (solid) along with the effective transfer function (dashed) corresponding to the forward and backward application of the filter. The filter cutoffs are 0.025 hr^{-1} (solid) and 0.02 hr^{-1} (dashed).

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

$$y(t_n) = b_0x(t_n) + b_1x(t_{n-1}) + b_2x(t_{n-2}) - a_1y(t_{n-1}) - a_2y(t_{n-2}) \quad (8)$$

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 $f < 0.0104 \text{ hr}^{-1}$ (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 proceeds and occasionally (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 τ 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' statement within FILTER_NAMES. For example, a suffix may need to be adjusted depending on the input file (*e.g.* '.seacor', '.despikes-prs', '.despikes-tmp').

Array bints

bints - The b-intercepts for calibrating τ 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

```

1 C****
2 C*****
3 C****
4 C**** PROGRAM: BUNS_APR88.FOR
5 C**** PURPOSE: To replace the original CARP and BUNS programs. Reads
6 C**** the data file created from the cassette reader and interprets
7 C**** the message codes. The good data bits are then decoded into
8 C**** 32-bit computer words.
9 C****
10 c00000 Revised May 1989 BUNS_MAY89.FOR
11 c00000 Includes the following steps for plotting the data:
12 c00000 a) writes only every 4th sample to unformatted plot files
13 c00000 b) 10 lbs of tau and seqno are written to separate files
14 c00000 in addition to the full values
15 c00000
16 C****
17 C**** I/O UNITS: KREAD = 7 --> Input data
18 C**** KCTRL = 8 --> Input control card file
19 C**** KLOG = 9 --> Output log file
20 C**** KWRITE = 10 --> Output data (all types)
21 C**** KPLOTS = 11 --> Output PLOTS data (TT's only)
22 C**** KSEQ = 12 --> Output sequence numbers for PLOTS
23 C**** KPRS = 13 --> Output pressure data for PLOTS
24 C**** KTMP = 14 --> Output temperature data for PLOTS
25 C*****
26 c00000 Revised AUG 1989 BUNS_AUG89.FOR
27 c**** Modified from buns_may89 so that no "junk" was inserted at the end
28 c**** of the buns plots.
29 c****
30 INTEGER*4 OUTREC(100), SAMPLING_PERIOD, mask, lsb(100)
31 INTEGER*4 KPLOTS, KSEQ, KPRS, KTMP, klsb, kslsb
32 INTEGER*2 INUM, NWDS, NUMBIT
33 INTEGER*2 ISHIFT(10), OUTWDS
34 INTEGER*4 TESTW(20), TESTV(20)
35 INTEGER*4 IW, OW, IT, LASTWD, IPTAU, IPSEQ, IPPRS, IPTMP
36 INTEGER*4 DECODE(200), NCASREC
37 INTEGER*4 SPANA(5), SPANB(5)
38 INTEGER*4 FSTTAU, LSTTAU, SEQPLT, PRSPLT, TMPPLT
39 INTEGER*2 EOF
40 INTEGER*4 KREAD, KWRITE, KLOG, KCTRL, KFMT
41 INTEGER*2 NLCC, NOVFL, NPE, NSR, NLS
42 REAL*4 OUTTAU(4800), OUTSEQ(4800), OUTPRS(4800), OUTTMP(4800)
43 real*4 tau1sb(4800), seq1sb(4800)
44 COMMON /CARD1/ KREAD, KWRITE, KLOG, KCTRL, KFMT
45 COMMON /CARD1A/ KPLOTS, KSEQ, KPRS, KTMP
46 COMMON /CARD2/ NWDS, NSECT, LASTWD, DECODE
47 COMMON /CARD2A/ SEQPLT, FSTTAU, LSTTAU, PRSPLT, TMPPLT
48 COMMON /CARD3/ ITHROT, NUNSP, SPANA, SPANB, TESTW, TESTV, ISHIFT
49 COMMON /CARD6/ NCASREC, NLCC, NOVFL, NPE, NSR, NLS
50 DATA KREAD/7/, KCTRL/8/, KLOG/9/, KWRITE/10/, KFMT/0/, KPLOTS/11/

```



```

51     DATA KSEQ/12/, KPRS/13/, KTMP/14/
52     DATA NCASREC/0/, EOF/0/, IOUTREC/0/
53     DATA IPTAU/0/, IPSEQ/0/, IPPRS/0/, IPTMP/0/
54     data mask/255/, klsb/15/.ksl=b/16/,mask10/1023/
55     data itotal/0/, istot/0/
56     data nskip/4/
57 C*****
58 C***** Open the I/O units and files
59 C***** Wait until later to open the output dataset according to the
60 C***** desired output form of the file (formatted or binary).
61 C***** Also open the plot units, if needed, in the S/R CONTROL_CARDS.
62 C*****
63     OPEN(UNIT=kread,STATUS='OLD',BLOCKSIZE=512,
64     C     FORM='UNFORMATTED',CARRIAGECONTROL='NONE',
65     C     ACCESS='SEQUENTIAL',RECORDTYPE='VARIABLE')
66     OPEN(UNIT=kctrl,STATUS='OLD')
67     OPEN(UNIT=klog,STATUS='NEW',FORM='FORMATTED')
68 C*****
69 C***** Read the control card file
70 C*****
71     CALL CONTROL_CARDS
72 C*****
73     WRITE(*,42)
74     42  FORMAT(1X,/,,' PROGRAM IS RUNNING. PLEASE WAIT. RUNNING TIME IS
75     C     08-12 MINUTES',/)
76 C*****
77 C*****                               Main Loop
78 C***** Process the data for one sampling period at a time and then write it
79 C***** to the output data set. You may need to use more than one cassette
80 C***** record (NSECT) for each sampling period.
81 C***** Decode GOODBITS into NWDS output words in the array OUTREC.
82 C*****
83     10 CONTINUE
84     OW = 0
85     IW = 0
86     IT = 1
87     DO 50 SAMPLING_PERIOD = 1, NSECT
88 C*****
89 C***** Process the next cassette record for this sampling period.
90 C*****
91     CALL NEWCARP(EOF)
92     IF (EOF .NE. 0) GO TO 90
93     NCASREC = NCASREC + 1
94 C*****
95 C***** Decode the words in this section and save them in the output
96 C***** array. DECODE stores the word lengths of the data.
97 C***** A negative value in DECODE denotes the end of a
98 C***** cassette record (which usually means a sample period).
99 C*****
100    15 CONTINUE
101    IW = IW + 1
102    NUMBIT = DECODE(IW)
103    IF (NUMBIT .GE. 0) THEN
104    OW = OW + 1

```

```

105         OUTREC(OW) = NXTBIT(NUMBIT)
106         IF (IW .LE. LASTWD) GO TO 15
107         WRITE(KLOG,20) LASTWD, IW
108     20     FORMAT(/5X,'PROGRAM ERROR - IW EXCEEDS LASTWD AT STMT# 15',
109     ①     /5X,'LASTWD =',I4,' IW =',I4,5X,'RUN TERMINATED')
110         STOP 15
111     END IF
112 C*****
113 C***** Check the value of all the test words in this section.
114 C***** If any test fails - ignore all output words decoded
115 C***** thus far and start decoding for section 1 again.
116 C*****
117     25 CONTINUE
118     IF (TESTW(IT) .GT. 0) THEN
119         IF (TESTW(IT) .LE. OW) THEN
120             I = TESTW(IT)
121             IF (OUTREC(I) .NE. TESTV(IT)) THEN
122                 WRITE(KLOG,30) TESTW(IT), TESTV(IT), OUTREC(I), NCASREC
123     30     FORMAT(/5X,'BAD VALUE FOR OUTPUT WORD#',I4,
124     ①     ', TESTVALUE =',I6,', ACTUAL VALUE =',I6,
125     ①     ' - CASSETTE REC#',I4)
126                 GO TO 10
127             END IF
128             IT = IT + 1
129             GO TO 25
130         END IF
131     END IF
132 C*****
133 C***** Testing is finished.
134 C***** End the decoding loop - repeat NSECT times.
135 C*****
136     50 CONTINUE
137 C*****
138 C***** Unspan the words which spanned sections and throw out any
139 C***** word at position ITHROT.
140 C*****
141     IU = 1
142     I = 1
143     J = 1
144     IF (ITHROT .NE. 0 .OR. NUMSP .NE. 0) THEN
145     60     OUTREC(I) = OUTREC(J)
146         IF (J .NE. ITHROT) THEN
147             IF (J .EQ. SPANA(IU)) THEN
148     65             JJ = SPANB(IU)
149                 OUTREC(I) = OUTREC(JJ) + (OUTREC(J)*2**ISHIFT(IU))
150                 IU = IU + 1
151                 J = JJ
152             END IF
153     70             I = I + 1
154         END IF
155     75             J = J + 1
156                 IF (J .LE. LASTWD) GO TO 60
157     END IF
158 C*****

```

```

159 C***** This should finish one sampling period.
160 C***** Calculate the number of words to be written out to the disk
161 C***** and write out to the disk file using the desired format.
162 C*****
163     80 CONTINUE
164     OUTWDS = NWDS-MUNSP-MSECT-ITHROT+1
165     IOUTREC = IOUTREC + 1
166     IF (KFMT .EQ. 0) THEN
167         WRITE(KWRITE) (OUTREC(IO), IO=1,OUTWDS) ! Unformatted output
168     ELSE
169         WRITE(KWRITE,85) (OUTREC(IO), IO=1,OUTWDS) ! Formatted output
170     85     FORMAT(8I10)
171     END IF
172 C*****
173 C***** Store the various data measurements in different output arrays for
174 C***** plotting. When the arrays are full, dump them out on disk.
175 C***** Only write every 4th sampling period to plotting files
176 C*****
177     nskip = nskip + 1
178     if (nskip .lt. 4) go to 10
179     nskip = 0
180     IF (FSTTAU .NE. 0) THEN
181         DO 88 ITAU = FSTTAU, LSTTAU
182             IPTAU = IPTAU + 1
183             OUTTAU(IPTAU) = OUTREC(itau)
184             taulsb(iptau) = jiaand(outrec(itau),mask10)
185     88     CONTINUE
186
187     IF (IPTAU .EQ. 4800) THEN
188         itotal = itotal + iptau
189         WRITE(KPLOT5) OUTTAU
190         write(klsb) taulsb
191         IPTAU = 0
192     END IF
193 end if
194 IF (seqPLT .NE. 0) THEN
195     IPSEQ = IPSEQ + 1
196     OUTSEQ(IPSEQ) = OUTREC(SEQPLT)
197     lsbseq = jiaand(outrec(seqplt),mask10)
198     seqlsb(ipseq) = lsbseq
199     IF (IPSEQ .EQ. 4800) THEN
200         istot = istot + ipseq
201         WRITE(KSEQ) OUTSEQ
202         write(kslsb) seqlsb
203         IPSEQ = 0
204     END IF
205 end if
206 IF (PRSPLT .NE. 0) THEN
207     IPPRS = IPPRS + 1
208     OUTPRS(IPPRS) = OUTREC(26)
209     IF (IPPRS .EQ. 4800) THEN
210         WRITE(KPRS) OUTPRS
211         IPPRS = 0
212     END IF

```

```

213     END IF
214     IF (TMPPLT .NE. 0) THEN
215         IPTMP = IPTMP + 1
216         OUTTMP(IPTMP) = OUTREC(27)
217         IF (IPTMP .EQ. 4800) THEN
218             WRITE(KTMP) OUTTMP
219             IPTMP = 0
220         END IF
221     END IF
222 C*****
223 C***** Go get next chunk of data.
224 C*****
225         GO TO 10
226 C*****
227 C***** End of file condition, wrap things up. First dump what's stored
228 C***** in the OUTTAU array. Note that there may be "junk" at the end of
229 C***** this array, if less than 4800 new points were used.
230 C*****
231     90 CONTINUE
232 c*****-----
233 c***** beginning of modification made 13-Aug-89
234 c***** the output arrays are set to zero beyond the real data to avoid
235 c***** "junk" (referred to below) at the end of the plots.
236 c*****
237 do 500 i=iptau,4800
238     taulsb(i)=0.0
239     outtau(i)=0.0
240 500 continue
241
242 c*****
243 c***** note I claim ipseq,iprs and iptmp should be equal, and therefore
244 c***** enclose them in a single DO loop.
245 c*****
246 do 501 i=ipseq,4800
247     seqlsb(i)=0.0
248     outseq(i)=0.0
249     outprs(i)=0.0
250     outtmp(i)=0.0
251 501 continue
252
253 c*****
254 c***** end of modification made 13-Aug-89
255 c*****-----
256
257
258     IF (FSTTAU .NE. 0) then
259         WRITE(KPLOT5) OUTTAU
260         itotal = itotal + iptau
261         write(klsb) taulsb
262     End if
263     IF (SEQPLT .NE. 0) then
264         WRITE(KSEQ) OUTSEQ
265         istot = istot + ipseq
266         write(kslsb) seqlsb

```

```

267         end if
268         IF (PRSPLT .NE. 0) WRITE(KPRS) OUTPRS
269         IF (TMPPLT .NE. 0) WRITE(KTMP) OUTTMP
270         WRITE(KLOG,95) NCASREC, IOUTREC
271         95 FORMAT(' END OF DATA ENCOUNTERED'/
272         @          ' NUMBER OF CASSETTE RECORDS PROCESSED = ',I6/
273         @          ' NUMBER OF OUTPUT DATA RECORDS WRITTEN = ',I6)
274         WRITE(KLOG,100) NLCC, NOVFL, NPE, NSR, NLS
275         100 FORMAT(/' NUMBER OF DOUBLE LAST CHARACTERS = ',I6/
276         @          ' NUMBER OF OVERRUN FLAGS = ',I6/
277         @          ' NUMBER OF PARITY ERRORS = ',I6/
278         @          ' NUMBER OF SHORT RECORDS = ',I6/
279         @          ' NUMBER OF LOW SIGNALS = ',I6)
280         WRITE(klog,43) itotal,istot
281         43 FORMAT(1X,/,',      **** PROGRAM IS FINISHED!! **',//
282         @          ' itau = ', i10,'      iseq = iprs = itmp = ',i10)
283         STOP
284         END
285 C*****
286 C*****
287 C*****
288 C*****                      SUBROUTINES
289 C*****
290 C*****
291 C*****
292 C*****
293 C*****
294 C*****      SUBROUTINE NXTBIT
295 C*****
296 C***** Purpose :
297 C***** To translate a string of 'N' bit integer words
298 C***** one word at a time, into standard 32 bit integer word
299 C*****
300 C*****      Input :
301 C*****      A string of bits representing a string of integer
302 C*****      words of varying lengths
303 C*****
304 C*****      Output :
305 C*****      A 32-bit integer word containing one 'N' bit word
306 C*****      from the input string , padded to the left with binary
307 C*****      zeroes if necessary.
308 C*****
309 C*****
310 C*****      Usage :
311 C*****      A call to this entry point is of the form
312 C*****      IWORD = NXTBIT (NUMBITS) - where IWORD is
313 C*****      a 32-bit integer word which is to contain the next
314 C*****      NUMBITS bits from the bit string being processed.
315 C*****
316 C*****
317 C*****      Errors :
318 C*****      If NUMBITS is less than or equal to zero or
319 C*****      If NUMBITS is greater than 31
320 C*****      then IWORD is set to -1 (all binary ones)

```

```

321 C***** If the total of NUMBITS in all calls to NXTBIT      *
322 C***** exceeds IBITS in last call to INEXT, an error      *
323 C***** message will be printed.                            *
324 C*****                                                       *
325 C*****                                                       *
326 C*****                                                       *
327 C***** BEWARE      NXTBIT is declared as a function subroutine      *
328 C*****                                                       *
329 C*****
330      FUNCTION NXTBIT(NBITS)
331      IMPLICIT INTEGER*2 (A-Z)
332      INTEGER*2 MASKUSED(8)
333      BYTE IARRAY(256)
334      INTEGER*4 NXTBIT, ANS, NEXTPART
335      INTEGER*4 KREAD, KWRITE, KLOG, KCTRL, KFMT
336      COMMON /CARD1/ KREAD, KWRITE, KLOG, KCTRL, KFMT
337      COMMON /CARD4/ WINX, BINX, IARRAY
338      SAVE MASKNEG, MASKUSED
339      DATA MASKNEG/'00FF'X/
340      DATA MASKUSED/'FF7F'X, 'FF3F'X, 'FF1F'X, 'FF0F'X,
341      0          'FF07'X, 'FF03'X, 'FF01'X, 'FF00'X/
342 C*****
343 C***** Check for errors in number of bits to process.
344 C*****
345      IF (NBITS.LE.0) THEN          ! ERROR, RETURN -1
346          WRITE(KLOG,66)
347      66  FORMAT(' NBITS LESS THAN OR EQUAL TO 0 -- NXTBIT SET TO -1')
348          NXTBIT = -1
349          RETURN
350      ELSE IF(NBITS .GT. 31)THEN    ! ERROR, RETURN -1
351          WRITE(KLOG,71)
352      71  FORMAT('NBITS GREATER THAN 31 -- NXTBIT SET TO -1')
353          NXTBIT = -1
354          RETURN
355      END IF
356 C*****
357 C***** Initialize ANS to 0 and PART to the left-most bits of IARRAY(WINX)
358 C***** IARRAY(WINX) - Current 8-bit string to process.
359 C***** BINX - The number of bits of IARRAY(WINX) which have already
360 C*****      been used (don't want to use them again).
361 C***** BITWNT - Total number of bits needed to create the 32-bit word
362 C*****
363      ANS = 0          ! New word to decode
364      PART = IARRAY(WINX)
365 C*****
366 C***** First mask off the 8 MSB's that make PART negative.
367 C***** Then mask off any bits which have already been used.
368 C*****
369      PART = IAND(PART, MASKNEG)
370      IF (BINX .GT. 0) THEN
371          PART = IAND(PART, MASKUSED(BINX))
372      END IF
373      PART = IISFT(PART, BINX)
374      BITWNT = NBITS          ! Total bits needed

```

```

375 C*****
376 C***** See if there are enough bits in this word to get the
377 C***** full 32-bit word.
378 C*****
379     50 CONTINUE
380     IF (BITWNT .GT. 8-BINX) THEN
381 C*****
382 C***** Not enough bits - then use all of this word and then come back
383 C***** here to get more bits from the next word.
384 C*****
385         BITNOW = 8 - BINX
386         BITWNT = BITWNT - BITNOW
387         WINX = WINX + 1
388         BINX = 0
389 C*****
390 C***** More than enough bits - use only the bits needed.
391 C*****
392     ELSE
393         BITNOW = BITWNT
394         BITWNT = 0
395         BINX = BINX + BITNOW
396         IF (BINX .EQ. 8) THEN
397             BINX = 0
398             WINX = WINX + 1
399         END IF
400     END IF
401 C*****
402 C***** Now have some or all of the bits needed. Right justify them.
403 C*****
404     IF (BITNOW .LT. 8) THEN
405         NOWSHFT = 8 - BITNOW
406         PART = IISHFT(PART, -NOWSHFT)
407     END IF
408     NEXTPART = PART
409 C*****
410 C***** Shift bits already in ANS to the left to make room for new bits.
411 C***** Then 'OR' in the new bits.
412 C*****
413     ANS = JISHFT(ANS,(BITNOW))
414     ANS = IOR(ANS,NEXTPART)
415 C*****
416 C***** Are more bits needed?
417 C***** If yes: then get the next iarray word. If no: return ANS.
418 C*****
419     IF (BITWNT.GT.0) THEN
420         PART = 0
421         NEXTPART = 0
422         PART = IARRAY(WINX)
423         PART = IAND(PART, MASKNEG)
424         GO TO 50
425     END IF
426     NXTBIT = ANS
427     RETURN
428     END

```

```

429 C*****
430 C*****
431 C*****
432 C***** SUBROUTINE: NEWCARP
433 C***** PURPOSE: To read the Sea Data Reader data file. Then interpret and *
434 C***** remove the message code bits; keep only the data bits. Process *
435 C***** one cassette record at a time.
436 C*****
437 C*****
438 C*****
439 SUBROUTINE NEWCARP(EOF)
440 IMPLICIT INTEGER*2 (A-Z)
441 BYTE DATABITS(256), CODEBITS(256), GOODBITS(200)
442 INTEGER*4 KREAD, KWRITE, KLOG, KCTRL, KFMT, NCASREC
443 COMMON /CARD1/ KREAD, KWRITE, KLOG, KCTRL, KFMT
444 COMMON /CARD4/ WINX, BINX, GOODBITS
445 COMMON /CARD6/ NCASREC, NLCC, NOVFL, NPE, MSR, NLS
446 SAVE LAST_USED, LSTWD, IBLOCK, MASKNEG
447 DATA IBLOCK/0/, LSTWD/256/, LAST_USED/256/
448 DATA MASKTYPE/'0080'X/, TYPEA/'0000'X/, MASKLCC/'0001'X/
449 DATA MASKPE/'0080'X/, MASKTRACK/'000F'X/, MASKSHORT/'0040'X/
450 DATA MASKLOWSIG/'0020'X/, MASKOVFL/'0040'X/
451 DATA MASKNEG/'00FF'X/
452 DATA NLCC/0/, NOVFL/0/, NPE/0/, MSR/0/, NLS/0/
453 C*****
454 C***** Main Processing Loop
455 C***** The data file is a binary unformatted file of 512-byte blocks
456 C***** Read one block at a time.
457 C*****
458 10 CONTINUE
459 EOR = 0
460 IKEEP = 0
461 DO 12 L=1,200
462 GOODBITS(L) = 0
463 12 CONTINUE
464 IF (LAST_USED .EQ. 256) THEN
465 READ(KREAD, END=60, ERR=15)
466 6 (DATABITS(IN), CODEBITS(IN), IN=1,256)
467 15 IBLOCK = IBLOCK + 1
468 LAST_USED = 0
469 END IF
470 FRSTWD = LAST_USED + 1
471 DO 50 NWORD = FRSTWD, LSTWD
472 NOWHIBITS = CODEBITS(NWORD)
473 NOWHIBITS = IAND(NOWHIBITS, MASKNEG)
474 C*****
475 C***** Bits numbered 0 to 15 with LSB = 0 and MSB = 15.
476 C***** Determine if a data word (type A - low) or a
477 C***** message word (type B - hi) using bit 15.
478 C*****
479 TEST = IAND(MASKTYPE, NOWHIBITS)
480 20 CONTINUE
481 IF (TEST .EQ. TYPEA) THEN ! data word - keep the necessary bits
482 C*****

```



```

483 C***** Check to see if its the last data word of the cassette record.
484 C***** Zero out all bits except bit 8; If hi, then last character.
485 C*****
486         TEST = IAND(MASKLCC, NOWHIBITS)
487         IF (TEST .EQ. MASKLCC) THEN
488             IF (IKEEP .EQ. 0) THEN ! Make sure there aren't two in a row
489                 WRITE(KLOG,24) NWORD, IBLOCK, NCASREC
490                 24         FORMAT(' DOUBLE LCC at word = ',I6,
491                     @         ' of block = ',I6,' (NCASREC = ',I6,')')
492                 NLCC = NLCC + 1
493                 GO TO 50
494             END IF
495             EOR = 1 ! end of record encountered
496         END IF
497 C*****
498 C***** Check to see if there is an overrun flag, indicating at least one
499 C***** missed scan of data. Zero out all bits except bit 14.
500 C***** If hi, then overrun has occurred.
501 C*****
502         TEST = IAND(MASKOVFL, NOWHIBITS)
503         IF (TEST .EQ. MASKOVFL) THEN
504             WRITE(KLOG,25) NWORD, IBLOCK, NCASREC
505             25         FORMAT(' OVERRUN FLAG at word = ',I6,
506                 @         ' of block = ',I6,' (NCASREC = ',I6,')')
507             NOVFL = NOVFL + 1
508         END IF
509 C*****
510 C***** Keep the data dits , get rid of the code bits
511 C*****
512             IKEEP = IKEEP + 1
513             GOODBITS(IKEEP) = DATABITS(NWORD)
514 C*****
515 C***** Processing Type B - message words
516 C*****
517         ELSE
518             NOWLOWBITS = DATABITS(NWORD)
519             NOWLOWBITS = IAND(NOWLOWBITS, MASKNEG)
520 C*****
521 C***** Test for parity errors. If bit 7 is hi, errors occurred.
522 C*****
523         30         TEST = IAND(MASKPE, NOWLOWBITS)
524         IF (TEST .EQ. MASKPE) THEN ! Parity error occurred
525             TEST = IAND(MASKTRACK, NOWLOWBITS)
526             WRITE(KLOG,35) TEST, NWORD, IBLOCK, NCASREC
527             35         FORMAT(' PARITY ERROR = ',I6,' at word = ',I6,
528                 @         ' of block = ',I6,' (NCASREC = ',I6,')')
529             NPE = NPE + 1
530         END IF
531 C*****
532 C***** Test for a short record. If bit 6 is hi, then yes.
533 C*****
534         TEST = IAND(MASKSHORT, NOWLOWBITS)
535         IF (TEST .EQ. MASKSHORT) THEN ! Record was short
536             WRITE(KLOG,40) NWORD, IBLOCK, NCASREC

```

```

537     40      FORMAT(' SHORT RECORD at word = ',I6,
538           @      ' of block = ',I6,' (NCASREC = ',I6,')')
539           NSR = NSR + 1
540           IF (IKEEP .NE. 0) EOR = 1
541           END IF
542 C*****
543 C***** Test for low signal strength during record. If bit 5 is hi, then yes.
544 C*****
545           TEST = IAND(MASKLOWSIG, NOWLOWBITS)
546           IF (TEST .EQ. MASKLOWSIG) THEN ! Signal strength was weak
547           WRITE(KLOG,45) NWORD, IBLOCK, NCASREC
548           45      FORMAT(' WEAK SIGNAL encountered at word = ', I6,
549           @      ' of block = ',I6,' (NCASREC = ',I6,')')
550           NLS = NLS + 1
551           END IF
552 C*****
553 C***** Check to see if there is an overrun flag, indicating at least one
554 C***** missed scan of data. Zero out all bits except bit 14.
555 C***** If hi, then overrun has occurred.
556 C*****
557           TEST = IAND(MASKOVFL, NOWHIBITS)
558           IF (TEST .EQ. MASKOVFL) THEN
559           WRITE(KLOG,25) NWORD, IBLOCK, NCASREC
560           NOVFL = NOVFL + 1
561           END IF
562           END IF
563 C*****
564 C***** Finished interpreting this word. If EOR (end of cassette record),
565 C***** return to main program to decode into 32-bit computer words.
566 C*****
567           IF (EOR .NE. 0) THEN
568           LAST_USED = NWORD
569           WINX = 1
570           BINX = 0
571           RETURN
572           END IF
573           50 CONTINUE
574 C*****
575 C***** Finished with this block of data, get the next one.
576 C*****
577           LAST_USED = 256
578           GO TO 12
579 C*****
580 C***** End of data encountered.
581 C*****
582           60 CONTINUE
583           EOF = -1
584           WRITE(KLOG,65) IBLOCK
585           65 FORMAT('//5X,'END OF FILE ENCOUNTERED FOLLOWING IBLOCK # ',I5//)
586           RETURN
587           END
588 C*****
589 C*****
590 C*****

```

```

591     SUBROUTINE CONTROL_CARDS
592     INTEGER*2 IDI, IDS(5), NWDS
593     INTEGER*4 IVALS(10), NSECT
594     INTEGER*4 NEG1CT, DECODE(200), IW
595     INTEGER*4 TESTW(20), TESTV(20), IT
596     INTEGER*4 ITHROT, SPANA(5), SPANB(5), IU
597     INTEGER*4 LASTWD, NUNSP
598     INTEGER*4 FSTTAU, LSTTAU, SEQPLT, PRSPLT, TMPPLT
599     INTEGER*2 ISHIFT(10)
600     INTEGER*4 KREAD, KWRITE, KLOG, KCTRL, KFMT
601     INTEGER*4 KPLOT5, KSEQ, KPRS, KTMP
602     COMMON /CARD1/ KREAD, KWRITE, KLOG, KCTRL, KFMT
603     COMMON /CARD1A/ KPLOT5, KSEQ, KPRS, KTMP
604     COMMON /CARD2/ NWDS, NSECT, LASTWD, DECODE
605     COMMON /CARD2A/ SEQPLT, FSTTAU, LSTTAU, PRSPLT, TMPPLT
606     COMMON /CARD3/ ITHROT, NUNSP, SPANA, SPANB, TESTW, TESTV, ISHIFT
607     DATA IDS/'NW', 'WL', 'SV', 'US', 'WF'/
608     DATA IW/0/, NEG1CT/0/, LASTWD/0/
609     DATA TESTW/20*0/, TESTV/20*0/, IT/0/
610     DATA ITHROT/0/, SPANA/5*0/, SPANB/5*0/, IU/0/, NUNSP/0/
611     C*****
612     C***** Read in the control parameters and set error options.
613     C*****
614     WRITE(KLOG,10)
615     10 FORMAT(5X,' CONTROL CARDS FOR DECODING'/)
616     C*****
617     C***** Reading loop for the the control cards
618     C*****
619     20 CONTINUE
620     READ(KCTRL,22,END=65) IDI, (IVALS(I),I=1,10)
621     22 FORMAT(A2,3X,10I5)
622     WRITE(KLOG,24) IDI, (IVALS(I),I=1,10)
623     24 FORMAT(/5X,A2,2X,10I10)
624     C*****
625     C***** Check for a 'NUMBER OF WORDS' (NW) card.
626     C***** NSECT - The number of cassette records used to hold
627     C***** all the data from one sampling period. Usually this
628     C***** is one.
629     C***** NWDS - Number of words to be decoded, this does not
630     C***** include the -1 word.
631     C*****
632     IF (IDI .EQ. IDS(1)) THEN
633         NWDS = IVALS(1)
634         NSECT = IVALS(2)
635         IF (NSECT .EQ. 0) NSECT = 1
636     C*****
637     C***** Check for a word length (WL) card.
638     C***** Save the word lengths in array 'DECODE' - Ignore zero values
639     C***** -1 on this card flags the end of the cassette record.
640     C*****
641     ELSE IF (IDI .EQ. IDS(2)) THEN
642         DO 30 I = 1, 10
643             IF (IVALS(I) .NE. 0) THEN
644                 IF (IVALS(I) .LT. 0) NEG1CT = NEG1CT + 1

```

```

645         IW = IW + 1
646         DECODE(IW) = IVALS(I)
647         END IF
648     30     CONTINUE
649 C*****
650 C***** Check for a 'SPECIAL VALUE' (SV) control card.
651 C***** Save the testwords in TESTW and the testvalues in TESTV.
652 C***** Ignore negative and zero values for testwords.
653 C*****
654         ELSE IF (IDI .EQ. IDS(3)) THEN
655             DO 40 I = 1, 10, 2
656                 IF (IVALS(I) .GT. 0) THEN
657                     IT = IT + 1
658                     TESTW(IT) = IVALS(I)
659                     TESTV(IT) = IVALS(I+1)
660                 END IF
661     40     CONTINUE
662 C*****
663 C***** Check for an 'UNSPAN' (US) words control card.
664 C***** First value is word to be thrown out (0=none).
665 C***** Other pairs of values are words to be unspanned.
666 C***** Save the values in SPANA and SPANB. Ignore zero values.
667 C*****
668         ELSE IF (IDI .EQ. IDS(4)) THEN
669             ITHROT = IVALS(1)
670             DO 50 I=2,9,2
671                 IF(IVALS(I) .GT. 0) THEN
672                     IU = IU + 1
673                     SPANA(IU) = IVALS(I)
674                     SPANB(IU) = IVALS(I+1)
675                 END IF
676     50     CONTINUE
677 C*****
678 C***** Check for a 'WRITE FORMAT' (WF) control card for
679 C***** output data on unit KWRITE.
680 C***** IF 0 -> UNFORMATTED     IF 1 -> FORMATTED
681 C***** Open the output file accordingly.
682 C***** Save the word numbers of the first and last travel times for plotting.
683 C***** Open all plotting file units if needed.
684 C*****
685         ELSE IF (IDI .EQ. IDS(5)) THEN
686             KFMT = IVALS(1)
687             IF (KFMT .EQ. 0) THEN
688                 OPEN(UNIT=kwrite,STATUS='NEW',FORM='UNFORMATTED')
689             ELSE
690                 OPEN(UNIT=kwrite,STATUS='NEW',FORM='FORMATTED')
691             END IF
692             SEQPLT = IVALS(2)
693             IF (SEQPLT .NE. 0) then
694                 OPEN(UNIT=kseq, STATUS='NEW', FORM='UNFORMATTED')
695                 OPEN(UNIT=kslsb,STATUS='NEW',FORM='UNFORMATTED')
696             end if
697             FSTTAU = IVALS(3)
698             LSTTAU = IVALS(4)

```

```

699         IF (FSTTAU .NE. 0) then
700             OPEN(UNIT=kplot5, STATUS='NEW', FORM='UNFORMATTED')
701             OPEN(UNIT=klab,STATUS='NEW',FORM='UNFORMATTED')
702         end if
703         PRSPLT = IVALS(5)
704         IF (PRSPLT .NE. 0)
705             @ OPEN(UNIT=kprs, STATUS='NEW', FORM='UNFORMATTED')
706             TMPPLT = IVALS(6)
707             IF (TMPPLT .NE. 0)
708                 @ OPEN(UNIT=ktmp, STATUS='NEW', FORM='UNFORMATTED')
709 C*****
710 C***** If control card is none of the above then it is invalid.
711 C***** So print a message and terminate the run.
712 C*****
713         ELSE
714             WRITE(KLOG,55)
715             55  FORMAT(/5X,'PRECEEDING CONTROL CARD HAS AN INVALID I.D.',
716                 @  ' - RUN TERMINATED')
717             STOP 55
718         END IF
719         GO TO 20
720 C*****
721 C***** Last control card has been read; Set parameters.
722 C*****
723         65 CONTINUE
724         LASTWD = IW
725         NUNSP = IU
726 C*****
727 C***** If there are words to be unspanned, get the multiplicative
728 C***** values from the associated word lengths. Assume normal bit
729 C***** order, LSB's to the right (higher word #).
730 C***** Note that the DECODE array still contains the ITHROT words
731 C***** so that IWD is calculated to skip over these values.
732 C*****
733         IF(NUNSP .NE. 0) THEN
734             DO 70 I=1,NUNSP
735                 IWD = SPANB(I) + I
736                 ISHIFT(I) = DECODE(IWD)
737             70  CONTINUE
738         END IF
739 C*****
740 C***** Check for control card consistency.
741 C*****
742         75 CONTINUE
743         IF (NSECT .NE. NEG1CT) THEN
744             WRITE(KLOG,78) NSECT, NEG1CT
745             78  FORMAT(/5X,'NW CARD SPECIFIES',I5,
746                 @  ' SECTIONS BUT WL CARDS CONTAIN',I5,
747                 @  ' SECTION END MARKERS (NEG) - RUN TERMINATED')
748             STOP 78
749         END IF
750         80 CONTINUE
751         I = IW - NEG1CT
752         IF (NWDS .NE. I) THEN

```

```
753      WRITE(KLOG,82) NWDS, I
754      82  FORMAT(/5X,'NW CARD SPECIFIES',I5,
755          @  ' WORDS BUT WL CARDS CONTAIN',I5,
756          @  ' (NON-NEGATIVE NON-ZERO) WORDS - RUN TERMINATED')
757      STOP 82
758      END IF
759  C*****
760  C***** Control cards were okay. Return to main program.
761  C*****
762      85 CONTINUE
763      RETURN
764      END
```

3.2 PUNS_MAY88.FOR

```

1 C*****
2 C*****                PUNS_MAY88.FOR
3 C*****
4 C*****      THIS PROGRAM WILL PRODUCE HISTOGRAMS AND/OR LISTINGS
5 C***** OF TRAVEL TIME BURSTS OF A SPECIFIED RANGE OF SAMPLES.
6 C***** THE LISTINGS ARE EITHER IN INTEGER COUNTS OR CONVERTED TO THE
7 C***** DECIMAL EQUIVALENTS. THREE TYPES OF HISTOGRAMS CAN BE
8 C***** PRODUCED:  1) LEVEL-1 (L1) ARE INDIVIDUAL PLOTS OF EACH
9 C***** SAMPLING PERIOD.  2) LEVEL-2 (L2) ARE PLOTS OF GROUPS OF
10 C***** RECORDS, SAMPLED AT A GIVEN INTERVAL.  3) LEVEL-3 PRODUCES A
11 C***** HISTOGRAM OF ALL RECORDS WITHIN THE RANGE SPECIFIED.  A LEVEL-3
12 C***** PLOT IS PRODUCED EACH AUTOMATICALLY WHEN THE PROGRAM IS
13 C***** EXECUTED.
14 C*****      THE USER SPECIFIES THE TOTAL RANGE OF RECORDS TO PROCESS.
15 C***** IF LEVEL-2 PLOTS ARE TO BE MADE, THE USER MUST ALSO SPECIFY THE
16 C***** NUMBER OF RECORDS TO USE FOR A HISTOGRAM (GRPSIZ) AND THE
17 C***** NUMBER OF RECORDS TO SKIP BETWEEN CONSECUTIVE PLOTS (RATE).
18 C***** THE UPPER AND LOWER BOUNDS OF HISTOGRAMS ARE ALSO SPECIFIED BY
19 C***** THE USER, THUS ENLARGEMENTS OF A NARROWER RANGE CAN BE MADE.
20 C***** IF THERE ARE ADDITIONAL SENSORS, SUCHAS PRESSURE, THEIR VALUES
21 C***** IN COUNTS ARE PRINTED WHEN SAMPLES ARE LISTED.  IF THERE ARE 2
22 C***** TT DETECTORS, BOTH WILL BE PLOTTED.
23 C*****
24 C***** FORTRAN UNIT NUMBERS DESIGNATED AS FOLLOWS:
25 C*****      KR      (UNIT 5) CONTROL CARD INPUT FILE
26 C*****      KW      (UNIT 6) PRINTER OUTPUT LOG FILE OF HISTOGRAMS
27 C*****      KWDEC  (UNIT 7) FLOATING POINT OUTPUT OF SCALED DATA
28 C*****      KWINT  (UNIT 8) INTEGER OUTPUT OF SCALED DATA
29 C*****      KRBUNS (UNIT 9) INTEGER INPUT OF BUNS.REV82 DATA
30 C*****
31 C*****
32      NAMELIST/CARD1/ HEADR
33      NAMELIST/CARD2/ NTT, TTYPE
34      NAMELIST/CARD3/ NWORDS, LBURST, LBFST, RDFMT
35      NAMELIST/CARD4/ NSEN, SENSOR, SWDNO
36      NAMELIST/CARD5/ SF1, SF2
37      NAMELIST/CARD6/ LBWDA, UBWDA, LBWDB, UBWDB
38      NAMELIST/CARD7/ START, END, RATE, GRPSIZ, SEQINC
39      NAMELIST/CARD8/ OPTW
40      COMMON  FREQ,LCT,UCT,LBND,UBND,NUMLN,NUMPL,SF,RATE,GRPSIZ
41      COMMON  NUMBR/NTT,LBFST,LBLST,TTYPE
42      COMMON  UNIT/KR, KW, KWDEC, KWINT, KRBUNS, RDFMT
43      CHARACTER*60 HEADR
44      CHARACTER*3 TTYPE(2)
45      CHARACTER*2 SENSOR(3), OPTW(4), PR, TP, AM
46      CHARACTER*2 DE, INT, L1, L2
47      INTEGER*4 FREQ(55,6), FREQA(55), FREQB(55)
48      INTEGER*4 LCT(6), UCT(6), LCTA, LCTB, UCTA, UCTB
49      INTEGER*4 NTT
50      INTEGER*4 NWORDS, LBURST, LBFST, RDFMT
51      INTEGER*4 NSEN, SWDNO(3)
52      INTEGER*4 LBND(2), UBND(2), LBWDA, LBWDB, UBWDA, UBWDB

```

```

53     INTEGER*4 START, END, RATE, RATCTR, GRPSIZ, GRPEND, SEQINC
54     INTEGER*4 DESW, INSW, GRL1, GRL2
55     INTEGER*4 PRSN,TPSN,AMSN,PRWDNO,TPWDNO,AMWDNO
56     INTEGER*4 LEVEL1, LEVEL2, LEVEL3
57     INTEGER*4 TO, FROM
58     INTEGER*4 NUMLN(2), NUMPL(2), NUMPLA, NUMPLB
59     INTEGER*4 IN(100), OUT(100), RECIN, RECOU, SEQNO
60     INTEGER*4 PRESS, TEMP, AMBNS
61     INTEGER*4 BOTA, TOPA, BOTB, TOPB
62     REAL*4     SF1, SF2, SF(2)
63     REAL*4     DOUT(100), DLBWA, DUBWA, DLBWB, DUBWB
64     EQUIVALENCE (FREQ(1,1),FREQA(1)),(FREQ(1,2),FREQB(1)),
65     C           (LCT(1),LCTA),(LCT(2),LCTB),
66     C           (UCT(1),UCTA),(UCT(2),UCTB),
67     C           (LBWD(1),LBWA),(LBWD(2),LBWB),
68     C           (UBWD(1),UBWA),(UBWD(2),UBWB),
69     C           (NUMPL(1),NUMPLA),(NUMPL(2),NUMPLB),
70     C           (SF(1),SF1),(SF(2),SF2)
71     PARAMETER (DE='DE',INT='IN',L1='L1',L2='L2')
72     PARAMETER (PR='PR',TP='TP',AM='AM')
73     DATA KR/5/, KW/6/, KWDEC/7/, KWINT/8/, KRBUNS/9/
74     DATA RECIN/0/,RECOU/0/,SEQNO/-1/,RATCTR/0/
75     DATA DESW/0/,INSW/0/,GRL1/0/,GRL2/0/
76     DATA PRESS/-99/,TEMP/-99/,AMBNS/-99/,NTT/1/
77     DATA TTYPE/2*'  '/,SENSOR/3*'  '/,SWDNO/0,0,0/
78     DATA CPTN /4*'  '/
79 C*****
80 C***** SOME OF THE VARIABLES:
81 C***** GRPSIZ - THE NUMBER OF CONSECUTIVE BURSTS TO BE SAMPLED
82 C***** RATE   - THE NUMBER OF BURSTS SKIPPED BETWEEN GROUPS
83 C***** SEQUENCE NUMBER INCREMENT:
84 C*****           1 = 15 MIN SAMPLING
85 C*****           2 = 30 MIN SAMPLING
86 C*****           4 = 60 MIN SAMPLING
87 C*****           8 = 120 MIN SAMPLING
88 C*****
89 C***** TYPICALLY SF1=20480.0,SF2=20480.0
90 C***** LOWER AND UPPER BOUNDS ARE SPECIFIED IN TERMS OF COUNTS,
91 C***** HENCE, TO FIND ALLOWED RANGE IN ENGINEERING UNITS, DIVIDE
92 C***** COUNTS BY SCALE FACTOR: E.G. 202752./20480.= 9.9 SECONDS
93 C***** NTT - NUMBER OF DIFFERENT TRAVEL TIME DETECTORS (TTYPE) USED.
94 C***** NWORDS - NUMBER OF WORDS PER BUNS OUTPUT RECORD (INCLUDING
95 C***** SEQ#, LBURST,PRESS,TEMP,AMB)
96 C***** RDFMT - FORMAT OF INPUT DATASET:  IF 1 => FORMATTED READ
97 C*****                                           IF 0 => BINARY READ
98 C*****
99 C***** INITIALIZE COMMON VARIABLES
100 C*****
101     DO 16 I = 1, 6
102     DO 15 J = 1,55
103     FREQ(J,I) = 0
104     16 CONTINUE
105     UCT(I) = 0
106     LCT(I) = 0

```



```

107 16 CONTINUE
108 C*****
109 C*****
110     WRITE(*,42)
111     42 FORMAT(1X,/, ' THE PROGRAM IS NOW RUNNING!',/,
112             @' THIS MAY TAKE A FEW MINUTES SO SIT BACK AND RELAX.',/)
113 C*****
114 C***** READ AND PRINT THE CONTROL CARD INFORMATION
115 C*****
116     READ (KR,NML=CARD1)
117     READ (KR,NML=CARD2)
118     READ (KR,NML=CARD3)
119     READ (KR,NML=CARD4)
120     READ (KR,NML=CARD5)
121     READ (KR,NML=CARD6)
122     READ (KR,NML=CARD7)
123     READ (KR,NML=CARD8)
124 C*****
125 C***** OPEN THE INPUT BUNS DATA SET FOR READING DEPENDING ON THE
126 C***** FORMAT OF THE DATA
127 C*****
128     IF (RDFMT .EQ. 0) THEN
129         OPEN(UNIT=KRBUNS, STATUS='OLD', FORM='UNFORMATTED')
130     ELSE
131         OPEN(UNIT=KRBUNS, STATUS='OLD', FORM='FORMATTED')
132     END IF
133 C*****
134 C***** RESET VARIABLES, IF NECESSARY, TO MAKE SURE THEY ARE CORRECT
135 C*****
136     IF (END .LT. 1) END = 2**30
137     IF (GRPSIZ .LE. 0) GRPSIZ = 1
138     IF (SEQINC .LE. 0) SEQINC = 1
139 C*****
140 C***** SET OPTION SWITCHES FOR THE DESIRED OUTPUT TYPES
141 C*****
142     DO 17 I = 1, 4
143     IF (OPTN(I) .EQ. INT) INSW = 1
144     IF (OPTN(I) .EQ. DE) DESW = 1
145     IF (OPTN(I) .EQ. L1) GRL1 = 1
146     IF (OPTN(I) .EQ. L2) GRL2 = 1
147 17 CONTINUE
148     IF (WSEN .EQ. 0) GO TO 26
149     DO 25 I = 1, 3
150     IF (SENSOR(I) .NE. PR) GO TO 23
151     PRSN = 1
152     PRWDNO = SWDNO(I)
153     GO TO 25
154 23 IF (SENSOR(I) .NE. TP) GO TO 24
155     TPSN = 1
156     TPWDNO = SWDNO(I)
157     GO TO 25
158 24 IF (SENSOR(I) .NE. AM) GO TO 25
159     AMSN = 1
160     AMWDNO = SWDNO(I)

```

```

161     25 CONTINUE
162 C*****
163 C***** DETERMINE THE REMAINING CONTROLLING VARIABLES
164 C***** THESE ARE BASED ON THE TYPE OF HISTOGRAMS WANTED
165 C*****
166     26 GRPEND = RATE + GRPSIZ
167     LEVEL3 = (GRL1 + GRL2) * 2 + 1
168     LEVEL2 = (GRL1 + GRL2) * 2 - 1
169     LEVEL1 = GRL1
170     LBLST = LBFST + LBURST - 1
171     IF(WTT .GT. 1) LBLST = LBFST + (2*LBURST) - 1
172 C*****
173 C***** WRITE HEADER INFO TO LOG
174 C*****
175     WRITE(KW,301) HEADR
176     301 FORMAT('1',A60)
177     WRITE(KW,303) (TTYPE(I),I=1,WTT),(SENSOR(II),II=1,3)
178     303 FORMAT('0',' TYPES OF SENSORS USED: ',5(A4,2X))
179     WRITE(KW,305) TTYPE(1)
180     305 FORMAT('0',A4,' DETECTOR: ')
181     IF(WTT .GT. 1) WRITE(KW,307) TTYPE(2)
182     307 FORMAT('+',T60,A4,' DETECTOR: ')
183     WRITE(KW,309) SF1
184     309 FORMAT(6X,'SCALE FACTOR A = ',F16.5)
185     IF(WTT .GT. 1) WRITE(KW,311) SF2
186     311 FORMAT('+',T60,5X,'SCALE FACTOR B = ',F16.5)
187     WRITE(KW,313) LBWDA,UBWDA
188     313 FORMAT(6X,'LBWDA = ',I8,3X,'UBWDA = ',I8)
189     IF(WTT .GT. 1) WRITE(KW,315) LBWDB,UBWDB
190     315 FORMAT('+',T60,5X,'LBWDB = ',I8,3X,'UBWDB = ',I8)
191     WRITE (KW,317) START,END,RATE,GRPSIZ,SEQINC
192     317 FORMAT(/10X,'REC #',I6,' THRU ',I6,' WILL BE PROCESSED',
193     @ //10X,'SAMPLE RATE =',I6,5X,'GROUP SIZE =',I6,
194     @ //10X,'SEQUENCE NO. INC. = ',I6)
195     WRITE (KW,319) (OPTW(I),I=1,4)
196     319 FORMAT(/10X,'OPTIONS = ',4(A2,2X))
197 C*****
198 C***** FIGURE THE RANGE OF EACH GRAPH LINE & THE # OF LINES / GRAPH
199 C*****
200     DO 30 I = 1,WTT
201     UBRNGE = UBND(I) - LBND(I)
202     NUMPL(I) = UBRNGE / 50 + 1
203     NUMLN(I) = UBRNGE / NUMPL(I) + 1
204     30 CONTINUE
205 C*****
206 C***** CONVERT THE UPPER AND LOWER BOUNDS TO DECIMAL SECONDS
207 C*****
208     DLBWDA = LBWDA / SF1
209     DUBWDA = UBWDA / SF1
210     IF(WTT .EQ. 1) GO TO 35
211     DLBWDB = LBWDB / SF2
212     DUBWDB = UBWDB / SF2
213 C*****
214 C***** READ THE INPUT DATA FILE, CHECK FOR EOF, INCREMENT COUNTER

```

```

215 C*****
216     35 CONTINUE
217     CALL RDBUNS(NWORDS,IN)
218     IF (IN(1) .EQ. -1) GO TO 1000
219     RECIN = RECIN + 1
220 C*****
221 C***** CHECK WHETHER RECORD SHOULD BE PROCESSED
222 C***** A) OUTSIDE RANGE OF RECORDS TO PROCESS
223 C*****
224     IF (RECIN .LT. START) GO TO 35
225     IF (RECIN .GT. END) GO TO 1100
226 C*****
227 C***** B) DOING LEVEL-1 OR LEVEL-3 PLOTS, USE THIS RECORD
228 C*****
229     IF (RATE .LT. 1) GO TO 40
230 C*****
231 C***** C) DOING ONLY A LEVEL-2 PLOT, CHECK IF WITHIN GROUP
232 C***** TO PROCESS OR TO SKIP. IF IN GROUP TO PROCESS,
233 C***** DO YOU HAVE THEM ALL? IF SO, THEN RESET COUNTER.
234 C*****
235     RATCTR = RATCTR + 1
236     37 IF (RATCTR .LT. RATE) GO TO 35
237     IF (RATCTR .LT. GRPEND) GO TO 40
238     RATCTR = RATCTR - RATE
239 C*****
240 C***** GENERATE A LEVEL-2 (GROUP) GRAPH IF REQUESTED
241 C*****
242     IF (GRL2.EQ.0) GO TO 37
243     TO = RECIN - 1
244     FROM = RECIN - GRPSIZ
245     CALL FREQGR (LEVEL2,FROM,TO)
246     GO TO 37
247 C*****
248 C***** CHECK FOR SEQUENCE ERRORS IN THE FILE
249 C***** AND RENAME THE DATA VALUES
250 C***** ASSUMES THAT SEQNO IS FIRST WORD AND THE TT'S ARE GROUPED
251 C*****
252     40 CONTINUE
253     IF (IN(1) .NE. SEQNO+SEQINC) WRITE(KW,335) RECIN,IN(1),SEQNO
254     335 FORMAT(/10X,'REC #',I6,' => SEQ #',I6,' RECORD OUT OF',
255     @ ' SEQUENCE (FORMER SEQ # WAS',I6,')')
256     SEQNO = IN(1)
257     IF(PRSW .EQ. 1) PRESS = IN(PRWDNO)
258     IF(TPSW .EQ. 1) TEMP = IN(TPWDNO)
259     IF(AMSW .EQ. 1) AMBNS = IN(AMWDNO)
260     DO 45 L=LBFST,LBLST
261     OUT(L)=IN(L)
262     45 CONTINUE
263 C*****
264 C***** IF REQUESTED, SCALE TTA AND TTB TO DECIMAL SECONDS AND PRINT
265 C*****
266     IF (DESW .EQ. 0) GO TO 50
267     DOUT(1) = SEQNO
268     DO 46 I = LBFST,LBLST,NTT

```

```

269     DOUT(I) = OUT(I) / SF1
270     IF(WTT .GT. 1) DOUT(I+1) = OUT(I+1) / SF2
271     46 CONTINUE
272     WRITE(KWDEC,410) RECIN,SEQNO,PRESS,TEMP,AMBNS,
273     * (DOUT(I),I=LBFST,LBLST)
274     410 FORMAT(5I10,/(8F10.5))
275     C*****
276     C***** IF REQUESTED, PRINT THE INTEGER DATA
277     C*****
278     50 CONTINUE
279     IF (INSW .EQ. 0) GO TO 60
280     WRITE(KWINT,420) RECIN,SEQNO,PRESS,TEMP,AMBNS,
281     * (OUT(I),I=LBFST,LBLST)
282     420 FORMAT(5I10,/(8I10))
283     C*****
284     C***** DETERMINE THE FREQUENCY DISTRIBUTION OF THE DATA
285     C***** AND LIMIT THE RANGE
286     C*****
287     60 CONTINUE
288     BOTA = 0
289     TOPA = 0
290     BOTB = 0
291     TOPB = 0
292     DO 66 I = LBFST,LBLST,WTT
293     IF (OUT(I) .GT. LBWDA) GO TO 62
294     OUT(I) = LBWDA
295     DOUT(I) = DLBWDA
296     BOTA = BOTA + 1
297     62 IF (OUT(I) .LT. UBWDA) GO TO 64
298     OUT(I) = UBWDA
299     DOUT(I) = DUBWDA
300     TOPA = TOPA + 1
301     64 CONTINUE
302     INDX = (OUT(I) - LBWDA) / NUMPLA + 1
303     FREQA(INDX) = FREQA(INDX) + 1
304     66 CONTINUE
305     C*****
306     C***** IF MORE THAN ONE DETECTOR WAS USED, CALCULATE THE FREQUENCY
307     C***** DISTRIBUTION OF THE SECOND MEASUREMENTS. INCREMENT COUNTERS.
308     C*****
309     IF(WTT .EQ. 1) GO TO 76
310     LB1 = LBFST + 1
311     DO 74 I = LB1,LBLST,WTT
312     IF (OUT(I) .GT. LBWDB) GO TO 70
313     OUT(I) = LBWDB
314     DOUT(I) = DLBWDB
315     BOTB = BOTB + 1
316     70 IF (OUT(I) .LT. UBWDB) GO TO 72
317     OUT(I) = UBWDB
318     DOUT(I) = DUBWDB
319     TOPB = TOPB + 1
320     72 CONTINUE
321     INDX = (OUT(I) - LBWDB) / NUMPLB + 1
322     FREQB(INDX) = FREQB(INDX) + 1

```

```

323 74 CONTINUE
324 LCTB = LCTB + BOTB
325 UCTB = UCTB + TOPB
326 C*****
327 C***** INCREMENT THE COUNTERS
328 C*****
329 76 RECOU = RECOU + 1
330 LCTA = LCTA + BOTA
331 UCTA = UCTA + TOPA
332 C*****
333 C***** GENERATE A LEVEL-1 (SINGLE RECORD) GRAPH IF REQUESTED
334 C*****
335 78 IF (GRL1 .EQ. 0) GO TO 35
336 FROM = RECIN
337 TO = 0
338 CALL FREQGR(LEVEL1, FROM, TO)
339 GO TO 35
340 C*****
341 C***** WRAP UP - WRITE OUT MESSAGES TO USER
342 C*****
343 1000 CONTINUE
344 WRITE(KW, 340)
345 340 FORMAT(/10X, 'PROCESSING ENDED AT END OF DATA')
346 1100 CONTINUE
347 WRITE(KW, 345) RECIN, RECOU
348 345 FORMAT(///T10, I8, ' LOGICAL RECORDS READ',
349 @//T10, I8, ' LOGICAL RECORDS PROCESSED')
350 C*****
351 C***** TO END:
352 C***** PRINT AND GRAPH THE FREQUENCY DISTRIBUTION
353 C***** GENERATE A LEVEL-3 (TOTAL) GRAPH
354 C*****
355 CALL FREQGR (LEVEL3, START, RECIN)
356 WRITE(*, 43)
357 43 FORMAT(1X, //, ' WE ARE NOW DONE! - GOOD LUCK!', /)
358 STOP
359 END
360 C*****
361 C*****
362 C*****
363 C***** SUBROUTINES
364 C*****
365 C*****
366 C*****
367 SUBROUTINE RDBUNS(NWORDS, IN)
368 COMMON/UNIT/KR, KW, KWDEC, KWINT, KRBUNS, RDFMT
369 INTEGER*4 IN(100), RDFMT
370 C*****
371 C***** READ EITHER BINARY OR FORMATTED DATA
372 C*****
373 IF(RDFMT .EQ. 1) GO TO 90
374 READ(KRBUNS, END=99)(IN(I), I=1, NWORDS)
375 RETURN
376 90 READ(KRBUNS, 95, END=99) (IN(I), I=1, NWORDS)

```

```

377     95 FORMAT(8I10)
378     RETURN
379     99 IN(1)=-1
380     100 CONTINUE
381     RETURN
382     END
383 C*****
384 C*****
385 C*****
386     SUBROUTINE FREQGR(LEVEL,FROM,TO)
387     COMMON FREQ,LCT,UCT,LBND,UBND,NUMLN,NUMPL,SF,RATE,GRPSIZ
388     COMMON/NUMBR/NTT,LBFST,LBLST,TTYPE
389     COMMON/UNIT/KR, KW, KWDEC, KWINT, KRBUNS, RDFMT
390     INTEGER*4 FREQ(55,6),LCT(6),UCT(6)
391     INTEGER*4 LBND(2),UBND(2),NUMLN(2),NUMPL(2)
392     REAL*4 SF(2)
393     INTEGER*4 RATE,GRPSIZ,TTYPE(2)
394     INTEGER*4 LEVEL,FROM,TO
395     INTEGER*2 LINE(110)
396     DATA LINE/110*'X'/
397 C*****
398 C***** GRAPH BOTH THE A AND B TRAVEL TIMES
399 C*****
400     DO 50 K = 1,NTT
401     L = LEVEL - 1 + K
402 C*****
403 C*****
404 C***** PRINT THE GRAPH HEADING
405 C*****
406     WRITE(KW,101) TTYPE(K),FROM
407 101 FORMAT(1H1/T50,A4,' FREQUENCY DISTRIBUTION',T100,'REC #',I6)
408     IF (TO .NE. 0) WRITE(KW,102) TO
409 102 FORMAT(1H+,T112,'THRU',I6)
410     IF (RATE .NE. 0) WRITE(KW,103) RATE,GRPSIZ
411 103 FORMAT(T100,'RATE ',I6,T112,'GRUP',I5)
412 C*****
413 C***** DETERMINE THE MAXIMUM VALUE TO BE GRAPHED
414 C*****
415     MAX = 0
416     DO 10 I = 1, 55
417 10 IF (FREQ(I,L) .GT. MAX) MAX = FREQ(I,L)
418 C*****
419 C***** PRINT THE GRAPH
420 C*****
421     NL = NUMLN(K)
422     DO 20 I = 1, NL
423     NOX = FREQ(I,L) * 100 / MAX + 1
424     IVAL = LBND(K) + (I-1) * NUMPL(K)
425     DVAL = IVAL / SF(K)
426     WRITE (KW,105) IVAL,DVAL,FREQ(I,L),(LINE(IX),IX=1,NOX)
427 105 FORMAT(I10,F10.5,I8,2X,102A1)
428 20 CONTINUE
429     WRITE (KW,110) LBND(K),LCT(L),UBND(K),UCT(L)
430 110 FORMAT (//T20,'# UNDER ',I8,' = ',I6,

```

```
431      @//T20,'# OVER ',I8,' = ',I6)
432 C*****
433 C***** ADD THE TOTALS FOR THIS LEVEL TO THE TOTALS FOR THE NEXT LEVEL
434 C***** AND ZERO THE TOTALS FOR THIS LEVEL
435 C*****
436      IF (L .GT. 4) GO TO 40
437      J = L + 2
438      DO 30 I = 1, 55
439      FREQ(I,J) = FREQ(I,J) + FREQ(I,L)
440      30 FREQ(I,L) = 0
441      UCT(J) = UCT(J) + UCT(L)
442      LCT(J) = LCT(J) + LCT(L)
443      UCT(L) = 0
444      LCT(L) = 0
445      40 CONTINUE
446      50 CONTINUE
447      RETURN
448      END
```

3.3 FILL_JAN91.FOR

```

1 c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 c%%%%%
3 c%%%%% fill_jan91.for
4
5 c***** Revision of FILL_AUG90.FOR by K. Tracey January 1991
6 c***** Reworked the code for handling bad/missing records:
7 c***** When good records are interspersed between bad and missing records,
8 c***** earlier versions of the code did not keep these yearhours in their
9 c***** correct order. In the earlier codes, all missing records were added
10 c***** at once; thus causing a good yearhour to be put out of sequence.
11 c***** Now the code has been modified to add the missing records before and
12 c***** after the good yearhours, keeping them in their correct order.
13 c%%%%%
14 c%%%%% revision of fill_jul88.for
15 c%%%%% This version will make three output files: travel time, temperature,
16 c%%%%% and pressure. Each will be assigned the proper time according to
17 c%%%%% the particular model PIES, URI or Sea Data. The motivation for
18 c%%%%% separating the records arose when it was found that the different
19 c%%%%% model PIES' didn't sample identically. The sampling relative to the
20 c%%%%% travel time is:
21 c%%%%% URI- temp -115 sec SD- temp 773.750 sec
22 c%%%%% press -115 sec press 1645.625 sec
23 c%%%%%
24 c%%%%% here the time represents the period AFTER the center of the travel
25 c%%%%% time measurement (center of the burst of 24 pings) when, for a given
26 c%%%%% scan the, particular sensor is sampled.
27 c%%%%%
28 c%%%%% A namelist was added to the control file which specifies if the
29 c%%%%% instrument is a pressure instrument, and if so what model. "card3"
30 c%%%%% has variables "pies" and "model". The first character of "pies" is
31 c%%%%% checked for a "Y" or "y". The first character of model is checked
32 c%%%%% for a "S" or "s" to designate a SD from a URI model echo sounder.
33 c%%%%% The relative sample times above are correspondingly added to the
34 c%%%%% yearhour column of the inputted memod file and outputted to files
35 c%%%%% *.tmp, *.prs, and *.fill.
36 C%%%%%
37 C%%%%% additional i/o units:
38 C%%%%% kw2 (UNIT 15) - pressure output file
39 C%%%%% kw3 (UNIT 16) - temperature output file
40 c%%%%%
41 c%%%%% Changes will be mostly lower case, but some capitalization.
42 c%%%%% more later, Fields 4-Aug-90
43
44
45 C*****
46 C***** FILL.JULY88.For
47 C*****
48 C***** ORIGINALLY WRITTEN BY J. GUNN JANUARY 1980
49 C***** REVISED BY K. TRACEY SINCE 1981
50 C***** CONVERTED FOR VAX S. WOOD 1988
51 C*****
52 C***** THE PURPOSE OF THIS PROGRAM IS TO SEARCH THROUGH THE IES

```



```

53 C***** RECORD TO MAKE SURE THAT THE TIMES ARE INCREMENTING CORRECTLY.
54 C***** THERE ARE TWO TYPES OF ERRORS IN THE TIME BASE: 1) THE DATA
55 C***** RECORD FROM A SAMPLING PERIOD IS MISSING, AND 2) THE RECORD IS
56 C***** THERE BUT THE TIME ASSOCIATED WITH IT IS INCORRECT. IF A
57 C***** RECORD IS MISSING, A NEW ONE IS INSERTED WITH INTERPOLATED
58 C***** VALUES.
59 C***** THE USER CAN EITHER PROCESS THE TOTAL DATASET OR SEARCH
60 C***** THROUGH A SMALLER PORTION BY SPECIFYING NSTRAT AND NSTOP TO
61 C***** SELECT THE RECORDS. IF A DATA GAP GREATER THAN MAXDLT IS
62 C***** ENCOUNTERED, THE PROGRAM HALTS.
63 C*****
64 c%%%%%%%%%%
65 c%%%%% logical unit numbers have been changed to avoid using
66 c%%%%% units 5 and 6.
67 c%%%%%%%%%%
68 C***** I/O UNITS:
69 C***** KR (UNIT 17) - CONTROL PARAMETERS
70 C***** KW (UNIT 18) - USERS OUTPUT LOG
71 C***** KR1 (UNIT 19) - INPUT DATASET FROM MEMOD
72 C***** KW1 (UNIT 20) - OUTPUT DATA FILE
73 C*****
74 C*****
75 C*****
76 CHARACTER*60 HEADR
77 character*3 model,press
78 logical pies
79 INTEGER*4 RECIN,recadd, RECOU, RDOFF, MAXBAD
80 INTEGER*4 WALL,NADD,NBAD,KR,KW,KW1
81 INTEGER*4 FLAG,NFLAG,LSTREC,LOKREC
82 INTEGER*4 LINECT,LINPPG,LINADD
83 integer*4 index(300), need(300)
84 REAL*4 MAXDLT, DELTAT
85 REAL*4 TT,PR,TP,AM,YRHR
86 REAL*4 LOKTT,LOKPR,LOKTP,LOKAM,LOKYHR
87 REAL*4 LSTTT,LSTPR,LSTTP,LSTAM,LSTYHR
88 REAL*4 TTADD,PRADD,TPADD,AMADD,YHRADD
89 REAL*4 DLTT,DLTPR,DLTTP,DLTAM,DIFYHR
90 REAL*4 SAVTT(300),SAVPR(300),SAVTP(300),SAVAM(300)
91 real*4 SAVYHR(300)
92 REAL*4 OKDLT, IESDLT
93 real*4 goodyr(0:300),goodtt(0:300), goodpr(0:300), goodtp(0:300)
94 real*4 midokyr
95 PARAMETER (KR=17,KW=18,KR1=19,KW1=20,kw2=15,kw3=16)
96 PARAMETER (LINPPG=54)
97 NAMELIST/CARD1/ HEADR
98 NAMELIST/CARD2/ NSTART,NSTOP,MAXDLT,DELTAT
99 namelist/card3/ press,model
100 DATA NADD/0/,NBAD/0/,WALL/0/
101 DATA RECIN/0/,RECOU/0/,RECADD/0/
102 DATA MAXBAD/300/,LINECT/55/,LINADD/0/
103 DATA FLAG/-1/,pies/0/
104 C*****
105 C***** Open I/O units and files
106 C*****

```

```

107     OPEN(UNIT=KR,STATUS='OLD',FORM='FORMATTED',READONLY)
108     OPEN(UNIT=KW,STATUS='NEW',FORM='FORMATTED')
109     OPEN(UNIT=KR1,STATUS='OLD',FORM='FORMATTED',READONLY)
110     OPEN(UNIT=KW1,STATUS='NEW',FORM='FORMATTED')
111 C*****
112 C*****
113 C*****  READ THE CONTROL PARAMETERS AND WRITE TO LOG
114 C*****
115     READ(KR,WML=CARD1)
116     READ(KR,WML=CARD2)
117     READ(KR,WML=CARD3)
118 if ((press(1:1).eq.'y').or.(press(1:1).eq.'Y')) then
119 pies=.true.
120 open(unit=kw2,status='new',form='formatted')
121 open(unit=kw3,status='new',form='formatted')
122
123 if ((model(1:1).eq.'s').or.(model(1:1).eq.'S')) then
124 tmp_tcf= 1645.645/3600.
125 prs_tcf= 773.750/3600.
126 else
127 tmp_tcf=-115./3600.
128 prs_tcf=-115./3600.
129 endif
130 endif
131 WRITE(KW,410) HEADR
132 410 FORMAT(//A60)
133 WRITE(KW,415) NSTART,NSTOP,MAXDLT,DELTAT
134 415 FORMAT(//5X,'NSTART =',I5,9X,'NSTOP =',I5,9X,'MAXDLT(HRS)=' ,
135          @F10.4,5X,'DELTAT =',F10.4)
136 C*****
137 C*****  RESET PARAMETERS
138 C*****
139     NGAP = MAXDLT/DELTAT + 0.5
140     IF (NGAP .LE. 300) GO TO 5
141     WRITE(KW,416)
142 416 FORMAT(' ***** MAXDLT TOO BIG FOR ARRAYS *****'/
143          @ ' RESET MAXDLT OR CHANGE DIMENSIONS'/
144          @ ' RUN TERMINATED')
145     STOP 416
146 5 CONTINUE
147     IF (MAXBAD .NE. NGAP) MAXBAD = NGAP
148 C*****
149 C*****  SKIP OVER INITIAL RECORDS IF DESIRED.
150 C*****  INCREMENT INPUT AND OUTPUT COUNTERS.
151 C*****
152     I=NSTART
153 10 IF(I .LE. 1) GO TO 20
154     READ(KR1,420,END=80) TT,PR,TP,AM,YRHR
155 420 FORMAT(5E15.7)
156     RECIN=RECIN + 1
157     WRITE(KW1,420) TT,YRHR
158 if (pies) then
159 write(kw2,420) pr,yrhr+prs_tcf
160 write(kw3,420) tp,yrhr+tmp_tcf

```

```

161 endif
162
163     RECOUT = RECOUT+1
164     I=I-1
165     GO TO 10
166 C*****
167 C***** BEGIN PROCESSING BY READING NEXT DATA RECORD.
168 C***** ASSUME ITS YEARHOUR IS CORRECT.
169 C*****
170     20 CONTINUE
171     READ(KR1,420,END=80) TT,PR,TP,AM,YRHR
172     RECIN=RECIN+1
173     WRITE(KW,425) RECIN,YRHR
174     425 FORMAT(/5X,'FIRST RECORD OF THE SERIES IS INPUT REC# = ',I5,
175     @5X,'YRHR = ',F12.5////)
176 C*****
177 C***** MAIN PROCESSING LOOP
178 C***** WRITE RECORD IF TIME IS GOOD
179 C***** SAVE VALUES AS 'LAST OKAY'
180 C*****
181     25 WRITE(KW1,420) TT,YRHR
182 if (pies) then
183 write(kw2,420) pr,yrhr+prs_tcf
184 write(kw3,420) tp,yrhr+tmp_tcf
185 endif
186
187     RECOUT=RECOUT+1
188     LOKYHR=YRHR
189     LOKTT=TT
190     LOKPR=PR
191     LOKTP=TP
192     LOKAM=AM
193 C*****
194 C***** SAVE THE MOST RECENTLY READ DATA VALUES
195 C***** CHECK FOR END OF PROCESSING
196 C*****
197     30 CONTINUE
198     IF(NBAD .GE. MAXBAD) GO TO 90
199     IF(RECIN .GE. NSTOP) GO TO 70
200     LSTYHR=YRHR
201     LSTTT=TT
202     LSTPR=PR
203     LSTTP=TP
204     LSTAM=AM
205 C*****
206 C***** READ NEXT DATA RECORD, CHECK FOR PROPER SEQUENCING
207 C*****
208     READ(KR1,420,END=80) TT,PR,TP,AM,YRHR
209     RECIN=RECIN+1
210     IESDLT=YRHR-LSTYHR
211 C*****
212 C***** A) THE SEQUENCING IS WRONG, SAVE THIS ONE AS BAD
213 C***** AND THEN GO GET THE NEXT RECORD
214 C*****

```

```

215     IF(ABS(IESDLT - DELTAT) .GT. 0.1) GO TO 35
216 C*****
217 C***** B) THIS ONE IS OKAY, BUT THE PREVIOUS RECORDS WERE BAD
218 C***** SO WORK ON CLEANING UP THE DATASET
219 C*****
220     IF(NBAD .NE. 0) GO TO 40
221 C*****
222 C***** C) THIS ONE IS OKAY, PREVIOUS WERE OKAY, SO GET NEXT RECORD
223 C*****
224     GO TO 25
225 C*****
226 C***** RECORD IS OUT OF SEQUENCE - SAVE IT
227 C*****
228     35 CONTINUE
229     NBAD = NBAD+1
230     SAVYHR(NBAD)=YRHR
231     SAVTT(NBAD)=TT
232     SAVPR(NBAD)=PR
233     SAVTP(NBAD)=TP
234     SAVAM(NBAD)=AM
235     GO TO 30
236 C*****
237 C***** PROPER SEQUENCING HAS RESUMED, BUT PREVIOUS RECORDS
238 C***** WERE OUT OF ORDER.
239 C*****
240     40 CONTINUE
241     DLTYHR=LSTYHR-LOKYHR
242     OKDLT=DLTYHR-NBAD*DELTAT
243     IF(OKDLT .GT. MAXDLT) GO TO 90
244     WADD=WADD+RDOFF(OKDLT/DELTAT)
245     NBAD=NBAD-1
246     WALL=NBAD+WADD
247 C*****
248 C***** WRITE TO LOG - OUT OF SEQUENCE RECORDS
249 C*****
250     IF(LINECT .LT.LINPPG) GO TO 50
251     WRITE(KW,430)
252     430 FORMAT('1')
253     WRITE(KW,432)
254     432 FORMAT('//7X,'LAST GOOD SEQUENCING',8X,'SEQUENCING RESUMED',
255     @6X,'# ADDED',5X,'RECORDS WITH YRHR'/
256     @9X,'RECIN',4X,'LSTOK YRHR',8X,'RECIN',5X,'LST YRHR',
257     @6X,'RECORDS',6X,'OUT OF SEQUENCE')
258     WRITE(KW,434)
259     434 FORMAT('+',2(5X,'-----'),2(5X,'-----'),
260     @'-----'//)
261     LINECT=0
262     50 CONTINUE
263     LOKREC=RECIN-NBAD-2
264     LSTREC=RECIN-1
265     IF(NBAD .EQ.0) GO TO 52
266     WRITE(KW,435) LOKREC,LOKYHR,LSTREC,LSTYHR,WADD,(SAVYHR(I),
267     @I=1,NBAD)
268     435 FORMAT(1X,2(5X,I10,2X,F10.2),5X,I7,5X,5F10.2,10(/72X,5F10.2))

```

```

269     IF(MOD(NBAD,5) .NE. 0) LINADD=1
270     LINADD=LINADD+1
271     LINECT=LINECT+LINADD
272     LINADD=0
273     GO TO 55
274     52 CONTINUE
275     WRITE(KW,435) LOKREC,LOKYHR,LSTREC,LSTYHR,NADD
276     LINECT=LINECT+1
277     C*****
278     C***** IF MISSING RECORDS, determine if any of the "bad" records may
279     c ***** in fact be good. We will want to use them if possible, and add
280     c***** missing records before and/or after them as necessary
281     C*****
282     55 IF (NADD .gt. 0) then
283         nparts = 1
284         index(1) = nbad
285         midokyr = lokyhr
286         midindex = 0
287         nleft = nadd
288         goodyr(0) = lokyhr
289         goodtt(0) = loktt
290         goodpr(0) = lokpr
291         goodtp(0) = loktp
292
293         do 56 k = 1,nbad
294             if (savyhr(k) .gt. lokyhr .and. savyhr(k) .lt. lstyhr) then
295
296     c***** This may be a "good" yearhour. First make sure that it has proper
297     c***** incrementation.
298                 irem = int( (savyhr(k) - lokyhr) / deltat)
299                 have = lokyhr + irem*deltat
300
301     c***** Yes this is a good yearhour, determine how many records must be added
302     c***** before this one.
303                 if ( abs(have - savyhr(k)) .lt. 0.05 ) then
304                     diff = savyhr(k) - midokyr
305                     nwant = rdoff(diff/deltat)
306                     nhave = k - midindex
307                     nowneed = nwant - nhave
308                     if (nowneed .le. nleft) then
309                         if (nowneed .lt. 0) then
310                             if (nowneed + need(nparts-1) .eq. 0) then
311                                 nleft = nleft + need(nparts-1)
312                                 nparts = nparts - 1
313                                 nowneed = 0
314                             else
315                                 go to 56
316                             end if
317                         end if
318                     need(nparts) = nowneed
319                     nleft = nleft - need(nparts)
320                     index(nparts) = k
321                     goodyr(nparts) = savyhr(k)
322                     goodtt(nparts) = savtt(k)

```

```

323             goodpr(nparts) = savpr(k)
324             goodtp(nparts) = savtp(k)
325             midokyr = savyhr(k)
326             midindex = k
327             nparts = nparts + 1
328         end if
329     end if
330 end if
331 56 continue
332     need(nparts) = nleft
333     goodyr(nparts) = lstyhr
334     goodtt(nparts) = lsttt
335     goodpr(nparts) = lstpr
336     goodtp(nparts) = lsttp
337 end if
338 C*****
339 C***** INTERPOLATE IF NECESSARY AND WRITE OUT ALL 'SAVED' RECORDS
340 C*****
341     60 CONTINUE
342 c*****
343 c***** Case 1: No records missing, but some yearhours were bad.
344 c***** Fix Up: Adjust the yearhours to be correct; don't add any records.
345
346     if (nadd .eq. 0) then
347         do 61 ii = 1, nbad
348             yhradd = ii*deltat + lokyhr
349             write(kw1,420) savtt(ii),yhradd
350             if (pies) then
351                 write(kw2,420) savpr(ii),yhradd+prs_tcf
352                 write(kw3,420) savtp(ii),yhradd+tmp_tcf
353             endif
354             recout = recout + 1
355             nflag = nflag + 1
356         61 continue
357
358     else
359
360 c***** Case 2: Records must be added during one or more sub-zones,
361 c***** delineated by "good" yearhours.
362 c***** Fix up: Added records before the "good" records if needed.
363 c***** Interpolating travel time, pressure, and temperature.
364
365     yhradd = lokyhr
366     do 69 k = 1, nparts
367         if (k .eq. 1) then
368             lfst = 1
369         else
370             lfst = index(k-1) + 1
371         end if
372         if (k .eq. nparts) then
373             last = nbad
374         else
375             last = index(k) - 1
376         end if

```

```

377
378 c***** Add missing records first.
379     if (need(k) .ne. 0) then
380         DLtyr= goodyr(k) - goodyr(k-1)
381         DLTTT= goodtt(k) - goodtt(k-1)
382         DLTPR= goodpr(k) - goodpr(k-1)
383         DLTPP= goodtp(k) - goodtp(k-1)
384         do 63 kadd = 1, need(k)
385             yhradd = deltat + yhradd
386             difyhr=kadd*deltat/dltyr
387             ttadd=difyhr*dlttt+loktt
388             pradd=difyhr*dltpr+lokpr
389             tpadd=difyhr*dlttp+loktp
390             write(kw1,420) ttadd,yhradd
391             if (pies) then
392                 write(kw2,420) pradd,yhradd+prs_tcf
393                 write(kw3,420) tpadd,yhradd+tmp_tcf
394             endif
395             nflag=nflag+1
396             recadd=recadd+1
397             nadd=nadd-1
398 63         continue
399     end if
400
401 c ***** Next right out any records with bad yearhours.
402
403     do 65 l = lfst, last
404         yhradd = deltat + yhradd
405         write(kw1,420) savtt(l),yhradd
406         if (pies) then
407             write(kw2,420) savpr(l),yhradd+prs_tcf
408             write(kw3,420) savtp(l),yhradd+tmp_tcf
409         endif
410         nflag=nflag+1
411         recout=recout+1
412 65     continue
413
414 c ***** Finally write out the "good" record if it lies between bad ones.
415
416     if (k .lt. nparts) then
417         write(kw1,420) savtt(index(k)),savyhr(index(k))
418         if (pies) then
419             write(kw2,420) savpr(index(k)),savyhr(index(k))+prs_tcf
420             write(kw3,420) savtp(index(k)),savyhr(index(k))+tmp_tcf
421         endif
422         recout = recout + 1
423         yhradd = savyhr(index(k))
424     end if
425 69     continue
426 end if
427 c ***** All finished with these records.
428
429
430     NADD=0

```

```

431      NBAD=0
432      WRITE(KW1,420) LSTTT,LSTYHR
433      if (pies) then
434      write(kw2,420) lstpr,lstyhr+prs_tcf
435      write(kw3,420) lsttp,lstyhr+tmp_tcf
436      endif
437
438      RECOU=RECOU+1
439      IF(RECOU .GE. NSTOP) GO TO 70
440      GO TO 25
441      C*****
442      C***** END OF PROCESSING - WRAP UP
443      C***** READ AND WRITE ANY REMAINING RECORDS
444      C*****
445      70 CONTINUE
446      READ(KR1,420,END=85) TT,PR,TP,AM,YRHR
447      RECIN=RECIN+1
448      WRITE(KW1,420) TT,YRHR
449      if (pies) then
450      write(kw2,420) pr,yrhr+prs_tcf
451      write(kw3,420) tp,yrhr+tmp_tcf
452      endif
453
454      RECOU=RECOU+1
455      GO TO 70
456      C*****
457      C***** UNEXPECTED END OF INPUT DATA
458      C*****
459      80 CONTINUE
460      IF(NBAD .NE. 0) GO TO 40
461      WRITE(KW,440)
462      440 FORMAT(/5X,'UNEXPECTED END OF FILE ENCOUNTERED BEFORE NSTOP',
463      @' RECORDS WERE READ')
464      C*****
465      C***** NORMAL END OF PROCESSING - WRITE TO USERS LOG
466      C*****
467      85 CONTINUE
468      WRITE(KW,442)RECIN,YRHR
469      442 FORMAT(///5X,'LST RECORD OF THE SERIES IS INPUT REC# = ',
470      @I5,5X,'YRHR = ',F12.5)
471      86 CONTINUE
472      RECOU=RECOU+RECADD
473      WRITE(KW,444) RECIN,RECADD,RECOU,NFLAG
474      444 FORMAT(/5X,'TOTAL RECORDS READ = ',T35,I10,
475      @/5X,'TOTAL RECORDS ADDED = ',T35,I10,
476      @/5X,'TOTAL RECORDS OUTPUT = ',T35,I10,
477      @/5X,'TOTAL FLAGGED YRHRS = ',T35,I10)
478
479      STOP
480      C*****
481      C***** TOO MANY OUT OF SEQUENCE RECORDS IN-A-ROW, TERMINATE RUN
482      C*****
483      90 CONTINUE
484      WRITE(KW,446) MAXBAD,RECIN

```



```
485 446 FORMAT(//5X,'MORE THAN ',I4,' CONSECUTIVE OUT-OF-SEQUENCE',
486 @' RECORDS ENCOUNTERED',
487 @//5X,'LAST INPUT RECORD READ WAS RECIN = ',I5,
488 @//5X,'RUN TERMINATED')
489 STOP 999
490 END
491 C*****
492 C*****
493 C*****
494 INTEGER FUNCTION RDOFF(REAL)
495 NUMBER=IFIX(REAL)
496 REST=REAL-NUMBER
497 IF(REST .LT. 0.5) GO TO 110
498 NUMBER=NUMBER+1
499 110 RDOFF=NUMBER
500 RETURN
501 END
```

3.4 MEMOD_JUL89.FOR

```

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

```

```

53 C*****
54     CHARACTER*60 HEADR
55     CHARACTER*4 IOPT(6), MED1, MED2, MOD1, MOD2, TT1, TT2
56     CHARACTER*3 TTYPE(2)
57     CHARACTER*2 SENSOR(3), PR, TP, AM
58     CHARACTER*2 EQN, TYPEQN(2), OVERNG, YES
59 C*****
60     INTEGER*4 KR, KW, KWDA, KWDB, KWLA, KWLB
61     INTEGER*4 NTT
62     INTEGER*4 NWORDS, LBURST, LBFST, RDFMT
63     INTEGER*4 NSEN, SWDNO(3)
64     INTEGER*4 NFIRST, NFSEQ, NLAST, NLSEQ, SEQINC
65     INTEGER*4 LBND1, UBND1, LBND2, UBND2
66     INTEGER*4 LAB, CTREF1, CTREF2
67     INTEGER*4 IX(100), IXB(100), IXX(100), IARRAY(100)
68     INTEGER*4 PWR2, PMID, TWOPWR(17)
69     INTEGER*4 PRWDNO, TPWDNO, AMWDNO
70     INTEGER*4 RECIN, SEQNO, ZERO, FOUR
71     INTEGER*4 IPASS, ISEQO, IPRESS, ITEMP, IAMBNS
72 C*****
73     INTEGER*2 PRCES1, PRCES2, TT1SW, TT2SW
74     INTEGER*2 PRSN, TPSN, AMSN
75     INTEGER*2 LINCTR, PGCTR, LINPPG
76     INTEGER*2 ENDFLG, OLD, NEW
77 C*****
78     REAL*4 XMED, RANGE1, RANGE2, AMSF, SF1, SF2
79     REAL*4 ARRAY(100), X1, X2, GRPHR
80     REAL*4 PRESS, TEMP, AMBNS, MTIME
81     REAL*8 GYRHR, DTIME, DGRPHR, OFFSET
82     REAL*8 AC1, AC2, AC3, BD1, BD2, BD3, T1, T2, T3, T4
83 C*****
84     PARAMETER(TT1=' TT1', TT2=' TT2')
85     PARAMETER (MED1 = 'MED1', MED2 = 'MED2', MOD1='MOD1', MOD2='MOD2')
86     PARAMETER(PR='PR', TP='TP', AM='AM')
87     PARAMETER(LINPPG=54, YES = 'YE')
88     PARAMETER(KR=5, KW=6, KWDA=7, KWDB=8, KWLA=9, KWLB=10)
89 C*****
90     COMMON/COMMED/ IARRAY
91     COMMON/COMMOD/ ARRAY
92     COMMON/MEMOCH/XMED, NGOODP, SDQRT, KT
93     COMMON/PARAM/NSKIP, NFIRST, LBURST
94     COMMON/UNIT/KRBUNS, RDFMT
95     COMMON/PCOEF/AC1, AC2, AC3, BD1, BD2, BD3, T1, T2, T3, T4
96     COMMON/TCOEF/TREF1, TREF2, CTREF1, CTREF2, TSEC, LAB
97     COMMON/PRSEQN/ EQN, OVERNG
98     COMMON/INDX/KF1, KL1, KF2, KL2
99 C*****
100     NAMELIST/CARD1/ HEADR
101     NAMELIST/CARD2/ NTT, TTYPE
102     NAMELIST/CARD3/ NWORDS, LBURST, LBFST, RDFMT
103     NAMELIST/CARD4/ NSEN, SENSOR, SWDNO
104     NAMELIST/CARD5/ SF1, SF2, AMSF
105     NAMELIST/CARD6/ NFIRST, NFSEQ, NLAST, NLSEQ, SEQINC
106     NAMELIST/CARD7/ LBND1, UBND1, LBND2, UBND2, DGRPHR

```

```

107     NAMELIST/CARD8/ IOPT
108     NAMELIST/CARD9/ IYR, MNTH, IDAY, I HOUR, MINUT, ISEC, ISEQO
109     NAMELIST/CARD10/ EQN,OVERNG
110     NAMELIST/CARD11/ AC1,AC2,AC3
111     NAMELIST/CARD12/ BD1,BD2,BD3
112     NAMELIST/CARD13/ T1,T2,T3,T4
113     NAMELIST/CARD14/ LAB,TSEC,TREF1,TREF2,CTREF1,CTREF2
114 C*****
115     DATA KRBUNS/11/
116     DATA TYPEQN/'AB','CD'/
117     DATA LINCTR/60/,PGCTR/0/
118     DATA TT1SW/0/,TT2SW/0/,PRCES1/0/,PRCES2/0/
119     DATA ENDFLG/0/, OFFSET/0.0/,IPASS/0/
120     DATA SF1/20480.0/,SF2/20480.0/,ZERO/0/,FOUR/4/
121     DATA RECIN/0/,IOPT/6*'  '/
122     DATA PRESS/-99./,TEMP/-99./,AMBWS/-99./
123     DATA SENSOR/3*'  ',SWDNO/0,0,0/
124     DATA PRSN/0/,TPSN/0/,AMSN/0/
125     DATA TREF1/0./,TREF2/0./,CTREF1/0/,CTREF2/0/
126     DATA LAB/0/,TSEC/0.0/, MIDOPT/0/
127     DATA TWOPWR/2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,
128     @          16384,32768,65536,131072/
129 C*****
130 C*****
131     write(*,42)
132     42 format(1X,/' PROGRAM IS RUNNING. PLEASE WAIT.'//,
133     @' HAVE SOME JAVA!!',//)
134 C***** OPEN THE CONTROL CARD FILE AND LOG FILE.
135 C***** THE I/O FILES WILL BE OPENED LATER AS NEEDED
136 C*****
137     OPEN(UNIT=KR, STATUS='OLD', FORM='FORMATTED', READONLY)
138     OPEN(UNIT=KW, STATUS='NEW', FORM='FORMATTED')
139 C*****
140 C***** READ THE CONTROL PARAMETERS FOR TYPES OF IES
141 C*****
142     READ(KR,NML=CARD1)
143     READ(KR,NML=CARD2)
144     READ(KR,NML=CARD3)
145     READ(KR,NML=CARD4)
146     READ(KR,NML=CARD5)
147     READ(KR,NML=CARD6)
148     READ(KR,NML=CARD7)
149     READ(KR,NML=CARD8)
150     READ(KR,NML=CARD9)
151 C*****
152 C***** READ IN CONTROL PARAMETERS FOR THOSE WITH ADDITIONAL SENSORS
153 C*****
154     IF(NSEN .NE. 0) THEN
155         READ(KR,NML=CARD10)
156         READ(KR,NML=CARD11)
157         READ(KR,NML=CARD12)
158         READ(KR,NML=CARD13)
159         READ(KR,NML=CARD14)
160     END IF

```

```

161 C*****
162 C***** CHECK FOR OVERRANGING OF EITHER TT1 OR TT2
163 C***** ASSUMES THAT ONLY ONE WILL OVERRANGE DURING A DEPLOYMENT.
164 C***** RESETS THE UPPER BOUND IF NECESSARY.
165 C*****
166   45 IF(UBND1 .LT. LBND1) THEN
167       MIDOPT=1
168       DO 500 LP=1,17
169       NOW2=TWOPWR(LP)
170       IF(NOW2 .GE. LBND1) THEN
171           PWR2=NOW2
172           PMID=TWOPWR(LP-1)
173           UBND1=UBND1+PWR2
174           GO TO 505
175       END IF
176   500 CONTINUE
177   END IF
178   505 CONTINUE
179   IF(UBND2 .LT. LBND2) THEN
180       MIDOPT=2
181       DO 510 LP=1,17
182       NOW2=TWOPWR(LP)
183       IF(NOW2 .GE. LBND2) THEN
184           PWR2=NOW2
185           PMID=TWOPWR(LP-1)
186           UBND2=UBND2+PWR2
187           GO TO 525
188       END IF
189   510 CONTINUE
190   END IF
191   525 CONTINUE
192 C*****
193 C***** WRITE OUT THE CONTROL PARAMETERS
194 C*****
195   WRITE(KW,310) HEADR
196   310 FORMAT(T20,' ***** MEMOD PROGRAM OUTPUT *****'/T10,A60//)
197   WRITE(KW,315) NTT,(TTYPER(I),I=1,2),NSEW,(SENSOR(I),I=1,3)
198   315 FORMAT(' THE FOLLOWING SENSORS ARE AVAILABLE: '/
199   @ I10,' TRAVEL TIME DETECTORS: ',2A4/
200   @ I10,' ADDITIONAL SENSORS: ',3(A2,2X)//)
201   WRITE(KW,320) NFIRST,NFSEQ,NLAST,NLSEQ,DGRPHR,SEQINC,LBURST
202   320 FORMAT(
203   @ ' RECORD #'S',I8,' (SEQ # ',I8,') THRU ',I8,'(SEQ #',I8,')',
204   @ ' WERE PROCESSED'// ' SAMPLING RATE IS ',D15.9,
205   @ ' DATA GROUPS PER HOUR(SEQINC = ',I5,')'//
206   @ ' ',I10,' DATA POINTS WERE USED FOR EACH MEDIAN VALUE'//)
207   WRITE(KW,330)LBND1,UBND1,LBND2,UBND2,(IOPT(J),I=1,6)
208   330 FORMAT(' PROCESSING PARAMETERS: '//
209   @ ' ',4X,'TT1MIN',4X,'TT1MAX',4X,'TT2MIN',4X,'TT2MAX'// ' ',4I10//
210   @ ' OPTIONS IN EFFECT = ',6(2X,A4)/
211   @ 10X,' NOTE: TT2 IS ONLY PROCESSED BY MEDIAN'//)
212   WRITE(KW,332) SF1,SF2,AMSF
213   332 FORMAT(' SCALING FACTORS FOR TT1, TT2, AND AMBS ARE: ',
214   @3F10.2//)

```

```

215     IF(MIDOPT .NE. 0) WRITE(KW,335) MIDOPT,PWR2,PMID
216 335 FORMAT(' TT DETECTOR #',I2,' WAS FOLDED TO AVOID',
217     @ ' WRAP-AROUND'/
218     @ 1X,I10,' WAS ADDED TO ALL POINTS LESS THAN', I10//)
219     IF(WSEN .EQ. 0) GO TO 48
220     IF(EQN .EQ. TYPEQN(1)) WRITE(KW,344)
221 344 FORMAT(' PAROS EQUATION USED: P = A(1-TO/T) - B(1-TO/T)**2')
222     IF(EQN .EQ. TYPEQN(2)) WRITE(KW,346)
223 346 FORMAT(' PAROS EQUATION USED: P = C{[1 - (TO/T)**2] - ',
224     @ 'D[1 - (TO/T)**2]**2}')
225     IF(OVERNG .EQ. YES) WRITE(KW,348)
226 348 FORMAT(' PRESSURE OVER-RANGED AT DEPTH 2**24 ADDED TO THE COUNTS')
227     WRITE(KW,341) AC1,AC2,AC3,BD1,BD2,BD3,T1,T2,T3,T4
228 341 FORMAT(' PRESSURE COEFFICIENTS: '//
229     @6X,'AC1',11X,'AC2',11X,'AC3'/3(1X,D12.5,1X)/
230     @6X,'BD1',11X,'BD2',11X,'BD3'/3(1X,D12.5,1X)/
231     @6X,'T1',12X,'T2',12X,'T3',12X,'T4'/4(1X,D12.5,1X)//)
232     WRITE(KW,342) TSEC,TREF1,TREF2,CTREF1,CTREF2
233 342 FORMAT(' SAMPLING TIME (SEC) FOR PRESS AND TEMP IS',F10.5//
234     @ ' CALIBRATION TEMPERATURES: ',2F10.5/
235     @ ' COUNTS: ',2I10)
236 C*****
237 C***** SET OPTION SWITCHES
238 C*****
239     48 DO 50 I=1,6
240     IF (IOPT(I) .EQ. MED1) THEN
241         PRCS1=1
242     ELSE IF (IOPT(I) .EQ. MED2) THEN
243         PRCS2=1
244     ELSE IF (IOPT(I) .EQ. MOD1) THEN
245         PRCS1=0
246     ELSE IF (IOPT(I) .EQ. MOD2) THEN
247         PRCS2=0
248     ELSE IF (IOPT(I) .EQ. TT2) THEN
249         TT2SW=1
250         OPEN(UNIT=KWDB, STATUS='NEW', FORM='FORMATTED')
251         OPEN(UNIT=KWLB, STATUS='NEW', FORM='FORMATTED')
252     ELSE IF (IOPT(I) .EQ. TT1) THEN
253         TT1SW=1
254         OPEN(UNIT=KWDA, STATUS='NEW', FORM='FORMATTED')
255         OPEN(UNIT=KWLA, STATUS='NEW', FORM='FORMATTED')
256     END IF
257 50 CONTINUE
258     IF(WSEN .NE. 0) THEN
259     DO 55 I = 1, 3
260     IF (SENSOR(I) .EQ. PR) THEN
261         PRSW = 1
262         PRWDNO = SWDNO(I)
263 53 ELSE IF (SENSOR(I) .EQ. TP) THEN
264         TPSW = 1
265         TPWDNO = SWDNO(I)
266 54 ELSE IF (SENSOR(I) .EQ. AM) THEN
267         AMSW = 1
268         AMWDNO = SWDNO(I)

```

```

269         END IF
270     55     CONTINUE
271         END IF
272 C*****
273 C***** OPEN THE BUNS INPUT DATA SET DEPENDING ON THE FORMAT
274 C*****
275         IF (RDFMT .EQ. 0) THEN
276             OPEN(UNIT=KRBUNS, STATUS='OLD', FORM='UNFORMATTED')
277         ELSE
278             OPEN(UNIT=KRBUNS, STATUS='OLD', FORM='FORMATTED')
279         END IF
280 C*****
281 C***** CALCULATE THE TIME BASE:
282 C***** TIME IS REFERENCED TO SEQUENCE NUMBER ISEQO, ASSUMES THAT
283 C***** TIME IS GIVEN AS GMT. THE MEDIAN TIME ASSOCIATED WITH
284 C***** ALL RECORDS IS OFFSET TO PLACE IT IN THE MIDDLE OF THE SAMPLE
285 C***** BURST
286 C***** WRITE OUT TIME BASE TO LOG.
287 C*****
288     65 CONTINUE
289         CALL YRDAY(IYR,MNTH,IDAY,IYRDAY)
290         CALL GMTYR(IYRDAY,IHOUR,MINUT,ISEC,GYRHR,ZERO,FOUR)
291         OFFSET=DFLOAT(LBURST-1)*10.0/7200.0
292         GYRHR = GYRHR + OFFSET
293         WRITE(KW,354) ISEQO,IYR,MNTH,IDAY,IHOUR,MINUT,ISEC
294     354 FORMAT('// TIME BASE PARAMETERS: '//
295             @' SEQUENCE NUMBER',I10,' IS ASSIGNED THE FOLLOWING TIME: '//
296             @8X,'IYR',6X,'MNTH',6X,'IDAY',5X,'IHOUR',5X,'MINUT',6X,'ISEC',
297             @/1X,6I10)
298         WRITE(KW,355) GYRHR,ISEQO
299     355 FORMAT('// GYRHR =',D15.9,' FOR ISEQO =',I10//)
300 C*****
301 C***** CALCULATE TOTAL NUMBER OF ECHOS AND RANGES OF THEM
302 C*****
303         KOUNT = LBURST * NTT
304         IF (TT1SW .NE. 0) THEN
305             RN1MIN = FLOAT(LBND1)
306             RN1MAX = FLOAT(UBND1)
307             RANGE1 = (RN1MAX - RN1MIN) /SF1
308         END IF
309         IF (TT2SW .NE. 0) THEN
310             RN2MIN = FLOAT(LBND2)
311             RN2MAX = FLOAT(UBND2)
312             RANGE2 = (RN2MAX - RN2MIN) /SF2
313         END IF
314 C*****
315 C*****     ENTER MAIN PROCESSING LOOP
316 C*****
317     6666 CONTINUE
318 C*****
319 C***** WRITE PAGE HEADING TO LOG IF NEEDED
320 C***** TYPE OF HEADER DEPENDS ON TT DETECTOR TYPE
321 C*****
322         IF (LINCTR .GE. LINPPG) THEN

```

```

323         LINCTR=0
324         PGCTR = PGCTR + 1
325     600     IF (TT1SW .NE. 0) THEN
326             IF (PRCES1 .NE. 1) THEN
327                 WRITE(KWLA,410) HEADR,PGCTR
328             ELSE
329     602         WRITE(KWLA,400) HEADR,PGCTR
330             END IF
331     604         WRITE(KWLA,415)
332     END IF
333     605     IF (TT2SW .NE. 0) THEN
334             IF (PRCES2 .NE. 1) THEN
335                 WRITE(KWLB,410) HEADR, PGCTR
336             ELSE
337     606         WRITE(KWLB,415) HEADR,PGCTR
338             END IF
339     608         WRITE(KWLB,415)
340     END IF
341 END IF
342     400     FORMAT(1H1,T10,A60,T110,'PAGE',I4, //T8,'SEQ#',T20,'MEDIAN',
343     ④ T32,'QUARTILE',T46,'#GOOD',T61,'XO',T69,'PRESS DBAR',
344     ④ T83,'TEMP (C)',T96,'AMB NOISE',T109,'TIME(MID)')
345     410     FORMAT(1H1,T10,A60,T110,'PAGE',I4, //T8,'SEQ#',T19,' MODE',
346     ④ T35,'SD',T46,'#GOOD',T61,'XO',T69,'PRESS DBAR',
347     ④ T83,'TEMP (C)',T96,'AMB NOISE',T109,'TIME(MID)')
348     415     FORMAT('+ ',9('-----',3X))
349 C*****
350 C***** READ IN THE DATA.
351 C***** INCREMENT COUNTER AND SET SEQNO
352 C*****
353     610 CONTINUE
354         CALL RDBUNS(NWORDS,IX)
355         RECIN = RECIN + 1
356         SEQNO=IX(1)
357 C*****
358 C***** SKIP UNWANTED RECORDS. IF FIRST ONE TO PROCESS,
359 C***** MAKE SURE THE SEQUENCE NUMBER IS THE ONE EXPECTED.
360 C*****
361         IF (IPASS .LE. 0) THEN
362             IF(RECIN .LT. NFIRST) GO TO 610
363             IF(SEQNO .EQ. NSEQ) GO TO 625
364             WRITE(KW,358) RECIN,SEQNO,NFSEQ
365     358     FORMAT(6X,'***** RUN TERMINATED *****',
366     ④ /' FOR RECORD # ',I8,' FIRST SEQNO WAS ',I8,' INSTEAD OF ',I8)
367             STOP 358
368         END IF
369 C*****
370 C***** CHECK FOR LAST RECORD TO BE PROCESSED
371 C***** WRITE A WARNING IF RECIN GETS BIGGER THAN EXPECTED
372 C*****
373     616 CONTINUE
374
375         IF (RECIN .EQ. NLAST) THEN
376             ENDFLG = 1

```



```

377     ELSE IF (RECIN .GE. NLAST) THEN
378     620   WRITE(KW,362)RECIN,NLAST,SEQNO
379     362   FORMAT(' WARNING: NLAST EXCEEDED! CHECK SEQUENCE NUMBERS. '/
380     @ ' RECIN = ',I10,' NLAST = ',I10,5X,' SEQNO = ',I10/
381     @ ' THIS GROUP HAS BEEN INCLUDED IN THE DATA SET'//
382     @ ' PROGRAM TERMINATES NORMALLY.'//)
383     ENDFLG = 1
384     END IF
385     C*****
386     C***** CHECK FOR END OF FILE FLAG. OTHERWISE, THIS RECORD
387     C***** IS TO BE PROCESSED, RENAME THE VARIABLES
388     C*****
389     625 CONTINUE
390     IF(SEQNO .EQ. -1) GO TO 7000
391     IF(PRSW .EQ. 1) IPRESS = IX(PRWDNO)
392     IF(TPSW .EQ. 1) ITEMP = IX(TPWDNO)
393     IF(AMSW .EQ. 1) IAMBWS = IX(AMWDNO)
394     DO 626 K=1,KOUNT
395     IXB(K)=IX(K+LBFST-1)
396     626 CONTINUE
397     C*****
398     C***** ASSIGN THE TIME TO THE MIDDLE OF THE GROUP INTERVAL
399     C*****
400     DTIME = GYRHR + (DFLOAT(SEQNO - ISEQO)/DGRPHR)/DFLOAT(SEQINC)
401     MTIME = DTIME
402     C*****
403     C***** WRITE OUT COUNTS OF FIRST SAMPLING PERIOD TO LOG FILE
404     C*****
405     IF(SEQNO .EQ. NFSEQ) WRITE(KW,360)SEQNO,DTIME,
406     @ (IXB(J),J=1,KOUNT)
407     360 FORMAT('//1H1,4X 'FIRST POINTS READ:'//' SEQNO = ',I10,
408     @ ' DTIME = ',E15.9/(4(2X,2I10)))
409     C*****
410     C***** CALCULATE THE REAL PRESSURE AND TEMPERATURE, IF NECESSARY
411     C*****
412     IF(PRSW .EQ. 1) CALL TEMPRS(IPRESS,ITEMP,PRESS,TEMP)
413     C*****
414     C***** CALCULATE THE AMBIENT NOISE, IF NECESSARY
415     C*****
416     IF(AMSW .EQ. 1) AMBWS = IAMBWS/AMSF
417     C*****
418     C***** IF OVERRANGING, WRAP EITHER TT1 OR TT2. WILL NOT DO BOTH
419     C*****
420     IF (MIDOPT .NE. 0) THEN
421     DO 630 J=MIDOPT,KOUNT,MTT
422     IF (IXB(J) .LT. PMID) IXB(J) = IXB(J) + PWR2
423     630 CONTINUE
424     END IF
425     635 CONTINUE
426     C*****
427     C***** ORDER THE DATA
428     C*****
429     CALL RESORT(KOUNT,MTT,IXB,IXX)
430     C*****

```

```

431 C***** PROCESS TT1 IF DESIRED
432 C***** IF 2 TT DETECTORS USED, PASS ONLY THE FIRST HALF OF IXX
433 C***** OTHERWISE ALL COUNTS WILL BE PASSED.
434 C***** CHECK WHETHER TO USE MODE OR MEDIAN PROCESS
435 C*****
436     IF (PRCES1 .EQ. 0) THEN
437         KT = 0
438         DO 640 K=KF1, KL1
439             KT = KT + 1
440             ARRAY(KT) = IXX(K)
441     640     CONTINUE
442         CALL TTMODE(SEQNO,RN1MIN,RN1MAX)
443     ELSE
444     642     CONTINUE
445         KT = 0
446         DO 644 K=KF1, KL1
447             KT = KT + 1
448             IARRAY(KT) = IXX(K)
449     644     CONTINUE
450         CALL TTMEDN(LBND1,UBND1)
451     END IF
452 C*****
453 C***** SCALE THE COUNTS TO SECONDS AND WRITE TO DISK AND LOG
454 C*****
455     646     X1 = XMED/SF1
456         WRITE(KWDA,420) X1,PRESS,TEMP,AMBNS,MTIME
457     420     FORMAT(5(2X,E13.7))
458         WRITE(KWLA,425)SEQNO,XMED,SDQRT,NGOODP,X1,PRESS,TEMP,AMBNS,MTIME
459     425     FORMAT(3X,I10,2(3X,F10.2),3X,I10,5(2X,F11.4))
460 C*****
461 C***** PROCESS TT2 IF DESIRED
462 C***** IF 2 TT DETECTORS PASS ONLY THE SECOND HALF OF IXX
463 C*****
464     650     IF (TT2SW .NE. 0) THEN
465         IF (PRCES2 .EQ. 0) THEN
466             KT = 0
467             DO 652 K=KF2,KL2
468                 KT = KT + 1
469                 ARRAY(KT) = IXX(K)
470     652     CONTINUE
471             CALL TTMODE(SEQNO,RN2MIN,RN2MAX)
472         ELSE
473             KT = 0
474             DO 656 K=KF2,KL2
475                 KT = KT + 1
476                 IARRAY(KT) = IXX(K)
477     656     CONTINUE
478             CALL TTMEDN(LBND2,UBND2)
479         END IF
480 C*****
481 C***** SCALE FROM COUNTS TO SECONDS, AND WRITE TO DISK AND LOG
482 C*****
483         X2 = XMED/SF2
484         WRITE(KWDB,420)X2,PRESS,TEMP,AMBNS,MTIME

```

```

485         WRITE(KWLB,425)SEQNO,XMED,SDQRT,NGOODP,X2,PRESS,TEMP,AMBNS,MTIME
486     END IF
487 C*****
488 C*****     END OF MAIN PROCESSING LOOP
489 C*****     INCREMENT THE COUNTERS, CHECK FOR END OF DATA
490 C*****
491         IPASS=IPASS+1
492         LINCTR = LINCTR + 1
493         IF (ENDFLG .EQ. 1) GO TO 7100
494         GO TO 6666
495 C*****
496 C***** UNEXPECTED END OF DATA TERMINATE PROGRAM
497 C*****
498     7000 CONTINUE
499         WRITE(KW,370)
500     370 FORMAT(/5X,' UNEXPECTED END OF DATA - RUN TERMINATED')
501 C*****
502 C***** NORMAL END OF PROCESSING WRITE MESSAGE TO LOG
503 C*****
504     7100 CONTINUE
505         WRITE(KW,375) SEQNO,DTIME,(IXB(J),J=1,KOUNT)
506     375 FORMAT(/1H0,4X,'LAST POINTS READ: '// SEQNO = ',I10,
507     @ ' DTIME = ',E15.9/(4(2X,2I10)))
508 C*****
509 C***** WRITE TT1 PROCESSING MESSAGES
510 C*****
511         IF (TT1SW .NE. 0) THEN
512             IF (PRCES1 .EQ. 0) THEN
513     7125         WRITE(KW,390) IPASS,TTYPER(1),RANGE1
514                 ELSE
515                 WRITE(KW,380) IPASS,TTYPER(1),RANGE1
516             END IF
517         END IF
518 C*****
519 C***** WRITE TT2 PROCESSING MESSAGES
520 C*****
521     7130 IF (TT2SW .NE. 0) THEN
522         IF (PRCES2 .EQ. 0) THEN
523     7132         WRITE(KW,390) IPASS,TTYPER(2),RANGE2
524                 ELSE
525                 WRITE(KW,380) IPASS,TTYPER(2),RANGE2
526             END IF
527         END IF
528     380 FORMAT(/5X,I10,' RECORDS WERE WRITTEN TO THE',A4,' MEDIUM DATASET
529     @'/ 15X,'RANGE OF DATA IS ',F10.5,' SEC')
530     390 FORMAT(/5X,I10,' RECORDS WERE WRITTEN TO THE',A4,' MODAL DATASET
531     @'/ 15X,'RANGE OF DATA IS ',F10.5,' SEC')
532     395 FORMAT(/5X,I10,' RECORDS WERE WRITTEN TO THE',A4,' MEDIUM DATASET
533     @'/ 15X,'RANGE OF DATA IS ',F10.5,' SEC')
534         WRITE(*,43)
535     43  FORMAT(1X,/, ' AH HA!! NOW YOU GOT TO DO SOME WORK!!'
536     @,/, ' I AM FINISHED - SO HAVE FUN!! ',/)
537     STOP
538     END

```

```

539 C*****
540 C*****
541 C*****
542 C*****          SUBROUTINES
543 C*****
544 C*****
545 C*****
546     SUBROUTINE RDBUNS(NWORDS,IN)
547     COMMON/UNIT/KRBUNS,RDFMT
548     INTEGER*4 IN(100),RDFMT
549 C*****
550 C*****     READ EITHER BINARY OR FORMMATED DATA
551 C*****
552     IF(RDFMT .EQ. 0) THEN
553         READ(KRBUNS,END=99)(IN(I),I=1,NWORDS)
554     ELSE
555         90 READ(KRBUNS,89,END=99) (IN(I),I=1,NWORDS)
556         89  FORMAT(8I10)
557     END IF
558     RETURN
559     99 IN(1)=-1
560     100 CONTINUE
561     RETURN
562     END
563 C*****
564 C*****
565 C*****
566     SUBROUTINE RESORT(KOUNT,MTT,XB,OUT)
567 C*****
568 C*****     NEW VERSION OF THE S/R ORDER
569 C*****     THIS PUTS ALL THE COUNTS FROM ONE DETECTOR NEXT TO EACH OTHER
570 C*****     AND PUTS THE COUNTS FROM THE SECOND ONE AFTER THOSE OF THE
571 C*****     FIRST. ONLY THEM ARE THE COUNTS FROM EACH DETECTOR SORTED
572 C*****     FROM LOW TO HIGH
573 C*****
574     COMMON/INDX/ KF1,KL1,KF2,KL2
575     INTEGER*4 OUT(226),TEMP,XB(226)
576 C*****
577 C*****     COPY XB TO OUT. KEEP TRACK OF INDEX OF FIRST AND LAST
578 C*****     TT FROM BOTH DETECTORS.
579 C*****
580     K = 0
581     DO 20 I=1,MTT
582         DO 10 J=I,KOUNT,MTT
583             K=K+1
584             OUT(K) = XB(J)
585         10 CONTINUE
586         KL1 = K
587         KF2 = K+1
588     20 CONTINUE
589     KF1 = 1
590     KL2 = KOUNT
591 C*****
592 C*****     SORT THE MEMBERS OF FIRST TT DETECTOR FIRST.

```

```

593 C***** THEN SORT THE SECOND TT MEMBERS.
594 C*****
595     LAST = KOUNT - NTT
596     DO 60 IN =1,NTT
597     IF (IN .GT. 1) GO TO 25
598     IFST = KF1
599     ILST = KL1
600     GO TO 26
601     25 CONTINUE
602     IFST = KF2
603     ILST = KL2
604 c*****
605 c***** test for removing no echos
606 c*****
607 26 dc 59  i=1,24
608 if ((out(i).eq.4351).or.(out(i).eq.4352)) out(i)=-out(i)
609 59 continue
610
611     DO 50 I = IFST, ILST
612     IF (OUT(I) .LE. OUT(I+1)) GO TO 50
613     J = I + 1
614     TEMP = OUT(J)
615     30 K = J - 1
616     IF (OUT(K) .LE. TEMP) GO TO 40
617     OUT(J) = OUT(K)
618     J = J - 1
619     IF (J .GT. IN) GO TO 30
620     40 OUT(J) = TEMP
621     50 CONTINUE
622     60 CONTINUE
623     RETURN
624     END
625 C*****
626 C*****
627 C*****
628     SUBROUTINE TTMEDN(RNMIN,RNMAX)
629 C*****     TAKES MEDIAN OF ARRAY XX.
630 C*****     FIRST WINDOWS ARRAY WITHIN RANGE = (RNMIN,RNMAX)
631 C*****     RETURNS NGOODP (# GOOD POINTS ) WITHIN RANGE
632 C*****     AND CALCULATES MEDIAN AND QUARTILE RANGE OF THOSE POINTS
633 C*****     ORIGINALLY WRITTEN SEPT 1978, A. CUTTING
634 C*****     REWRITTEN BY K. TRACEY JULY 1985: NOW ALL CALCULATIONS
635 C*****     ARE DONE AS INTEGERS, THEN PASSED BACK AS REALS.
636 C*****
637     COMMON/COMMED/ XX
638     COMMON /MEMOCH/ XMED, NGOODP, QUART, LBURST
639     INTEGER*4 RNMIN,RNMAX, XX(100)
640 C*****
641 C*****     ELIMINATE THE OUT OF RANGE RAW DATA POINTS
642 C*****
643     IBOT = 1
644     42 IF (XX(IBOT) .GE. RNMIN) GO TO 44
645     IBOT = IBOT + 1
646     IF (IBOT .LE. LBURST) GO TO 42

```

```

647      XMED = RMMIN
648      GO TO 47
649 C*****
650 C*****
651 C*****
652      44 CONTINUE
653      ITOP = LBURST
654      46 IF (XX(ITOP) .LE. RMAX) GO TO 48
655      ITOP = ITOP - 1
656      IF (ITOP .GT. IBOT) GO TO 46
657      XMED = RMAX
658 C*****
659 C***** ALL POINTS IN THE INTERVAL ARE OUT OF RANGE
660 C*****
661      47 CONTINUE
662      NGOODP = 0
663      QUART = 0.0
664      RETURN
665 C*****
666 C***** COMPUTE THE MEDIAN AND THE QUARTILE OF THE GOOD POINTS
667 C*****
668      48 CONTINUE
669      NGOODP = ITOP - IBOT + 1
670      INDMED = IBOT + NGOODP / 2
671      INDQLO = IBOT + (NGOODP + 2) / 4
672      INDQHI = INDQLO + (NGOODP - 1) / 2
673      XMED = XX(INDMED)
674      QUART = XX(INDQHI) - XX(INDQLO)
675      49 CONTINUE
676      RETURN
677      END
678 C*****
679 C*****
680 C*****
681      SUBROUTINE TEMPRS(PCT,TCT,PRS,TDEGC)
682 C*****
683 C***** EQUATIONS USING EITHER THE A, B OR THE C,D COEFFICIENTS.
684 C***** REVISED JULY 1985 - NOW INCLUDES THE POSSIBILITY OF
685 C***** USED TEMP (DEGC) TO CALCULATE PRESS FROM THE CD EQN.
686 C***** THIS IS DONE IF BD1 = BD2
687 C*****
688      INTEGER*4 PCT,TCT,CTREF1,CTREF2,LAB
689      INTEGER*2 EQN,OVERNG,YES,TYPEQN
690      REAL*4 PRS,TDEGC,TREF1,TREF2
691      REAL*8 ACCNST,AC1,AC2,BDCNST,BD1,BD2,TCONST,T1,T2,T,T3
692      COMMON/PCOEF/ACNST,AC1,AC2,BDCNST,BD1,BD2,TCONST,T1,T2,T3
693      COMMON/TCOEF/TREF1,TREF2,CTREF1,CTREF2,TSEC,LAB
694      COMMON/PRSEQN/ EQN,OVERNG
695      EQUIVALENCE (AC,A,C), (BD,B,D)
696      PARAMETER (TWO16=65536, TWO24=16777216)
697      PARAMETER(YES='YE',TYPEQN='CD')
698 C*****
699 C***** CHECK FOR OVERRANGING, THEN
700 C***** CALCULATE PERIOD (T) FROM PCT OF DATA SAMPLE

```

```

701 C***** WHERE: T = SAMPLING INTERVAL(IN SEC) / COUNTS
702 C*****
703 IF(OVERNG .EQ. YES) PCT=PCT+TWO24
704 c*****
705 c***** if a zero count is found set it to unity to prevent zero divide
706 c***** and make a easily distinguishable spike.
707 c*****
708 if (pct.eq.0) pct=1
709 T=TSEC/FLOAT(PCT)
710 C*****
711 C***** CALCULATE TEMPERATURE:
712 C***** INTERPOLATE FOR LAB TESTS: (T-T1)/(N-N1) = (T2-T1)/(N2-N1)
713 C***** IF NO LAB CALIBRATIONS, USE 'IDEAL' CONVERSION
714 C*****
715 IF(LAB .EQ. 0) TDEGC=FLOAT(TCT)*TREF1/TSEC
716 IF(LAB .EQ. 1) TDEGC=TREF1+FLOAT(TCT-CTREF1)*(TREF2-TREF1)/
717 float((CTREF2-CTREF1))
718 C*****
719 C***** IF BD1 = BD2, EQN IS CD AND COEF CALCULATED FROM TDEGC
720 C***** OTHERWISE:
721 C***** CALCULATE TEMP-DEPENDENT COEFFICIENTS A,B,TO
722 C***** AND C,D,TO FROM TDEGC
723 C***** THEN DETERMINE IF EQUATION IS AB OR CD TYPE
724 C*****
725 IF(BD1.EQ.BD2) GO TO 40
726 TDEGF=TDEGC*1.80+32.
727 AC=ACCNST+TDEGF*(AC1+AC2*TDEGF)
728 BD=BDCNST+TDEGF*(BD1+BD2*TDEGF)
729 TO=TCONST+TDEGF*(T1+T2*TDEGF)
730 IF(EQN.EQ.TYPEQN) GO TO 50
731 C*****
732 C***** CALCULATE PRESSURE FROM LINEARIZATION EQUATION:
733 C*****  $P=A(1-TO/T) - B(1-TO/T)**2$ 
734 C*****
735 TOT1=1-TO/T
736 PPSIA=TOT1*(A-B*TOT1)
737 GO TO 100
738 C*****
739 C***** TEMP-DEPENDANT COEFS ARE CALCULATED FROM TDEGC
740 C***** AND EQN IS AUTOMATICALLY CD TYPE
741 C*****
742 40 CONTINUE
743 AC=ACCNST+TDEGC*(AC1+AC2*TDEGC)
744 BD=BDCNST
745 TO=TCONST+TDEGC*(T1+T2*TDEGC+T3*TDEGC*TDEGC)
746 C*****
747 C***** CALCULATE PRESSURE FROM PAROS EQUATION:
748 C*****  $P=C\{[1-(TO/T)**2] - D[1-(TO/T)**2]**2\}$ 
749 C*****
750 50 CONTINUE
751 TOT=TO/T
752 TOTSQ1=1-TOT*TOT
753 PPSIA=C*(TOTSQ1 - D * TOTSQ1 * TOTSQ1)
754 100 CONTINUE

```

```

755 C*****
756 C*****      CONVERT TO DBAR FROM PSIA
757 C*****
758      PRS=PPSIA*0.68947
759      RETURN
760      END
761 C*****
762 C*****
763 C*****
764      SUBROUTINE TTMODE(LBLREC,RNMIN,RNMAX)
765 C***** DETERMINES MOST PROBABLE (MODAL) VALUE OF A RAYLEIGH DISTRI-
766 C***** BUTION  $P(X)=(X-XM)*EXP(-(((X-XM)**2)/2.*SD**2))$ 
767 C***** OF SAMPLE SIZE = NPTS ... BY THE METHOD OF MOMENTS, AFTER
768 C***** DESPIKING RELATIVE TO QUARTILE RANGE OF SAMPLE
769 C***** REVISED FROM TTMOD, SEPT 1978, R. WATTS
770 C***** REVISED 1 NOV 1982 BY KRL:
771 C***** WINDOWS THE DATA BASED ON MAX AND MIN SUPPLIED BY USER
772 C***** BEFORE DETERMINING THE QUARTILE RANGE.
773 C***** REVISED JULY 1985, NEW COMMON BLOCK INCLUDED AND CHOPS
774 C***** THE DATA BEFORE PASSING IT BACK TO MAIN PROGRAM
775 C*****
776      REAL*4 XM
777      REAL*4 XX(100),UPRLIM,LOWLIM
778      COMMON/COMMODO/ XX
779      COMMON/MEMOCM/XMOD,NGOODP,SD,NPTS
780      PARAMETER (KW=6)
781      INTEGER*4 UCNT
782      DATA XM/0.0/,UCNT/55/
783 C*****
784 C*****      INITIALIZE PARAMETERS. CHECK VALUE OF NPTS.
785 C*****
786      IF(NPTS.LT.8) GO TO 50
787      SM=0.
788      SD=0.0
789 C*****
790 C*****      ELIMINATE THE OUT OF RANGE POINTS BEFORE DETERMINING
791 C***** THE QUARTILE RANGE OF THE RAYLEIGH DISTRIBUTION
792 C*****
793      IBOT=1
794      5 IF(XX(IBOT) .GE. RNMIN) GO TO 10
795      IBOT=IBOT+1
796      IF(IBOT .LE. NPTS) GO TO 5
797      ngoodp=0
798      XMOD = RNMIN
799      GO TO 45
800      10 CONTINUE
801      ITOP=NPTS
802      15 IF(XX(ITOP) .LE. RNMAX) GO TO 20
803      ITOP=ITOP-1
804      IF(ITOP .GT. IBOT) GO TO 15
805      ngoodp=0
806      XMOD = RNMAX
807      GO TO 45
808      20 CONTINUE

```



```

809 c*****
810 c***** bin window the data using the routine binwindow. See documentation
811 c***** accompaning the code below
812 c*****
813 call binwindow(ibot,itop)
814 c*****
815 c***** if there are less than 4 points assign xmod=the upper puns limit.
816 c***** This will flag ignore statistically unreliable estimates as well as
817 c***** out of range ones.
818 c*****
819     NGOODP=ITOP-IBOT+1
820 if (ngoodp.lt.4) then
821     xmod=rnmax
822     goto 45
823 endif
824 C*****
825 C***** DETERMINE THE QUARTILE RANGE - NOTE:
826 C***** RAYLEIGH QUARTILE RANGE = .91 SIGMA
827 C***** FOR THE TTA DETECTOR THIS SHOULD BE APPROXIMATELY 2 MSEC
828 C*****
829     N75 = IBOT - 1 + nint(3.*float(NGOODP)/4.)
830     N50 = IBOT + NGOODP/2
831     N25 = IBOT - 1 + nint(float(NGOODP)/4.)
832     Q = XX(N75) - XX(N25)
833 C*****
834 C***** IF THE QUARTILE RANGE IS GREATER THAN 200. (APPROXIMATELY
835 C***** 10 MSEC) THEN THROW OUT THE WHOLE SAMPLE
836 C*****
837     IF (Q .GT. 200.) GO TO 40
838 C*****
839 C***** THROW OUT EVERYTHING OUTSIDE THE 97TH PERCENTILE RANGE
840 C*****
841     M=IBOT
842     K=ITOP
843 C*****
844 C***** DETERMINES UPRLIM AND LOWLIM FROM 97TH PERCENTILE RANGE
845 C***** ESTIMATED FOR RAYLEIGH DISTRIBUTION, BY THE RATIO TO
846 C***** QUARTILE RANGE
847 C*****
848
849 c*****
850 c***** the constants in this equation were changed from
851 c*****     UPRLIM = XX(N50) + 3.0* Q
852 c*****     LOWLIM = XX(N50) - 1.5* Q
853 c***** The new constants were found using the equations:
854 c***** variance=(2-pi/2)sigma^2
855 c***** mean     = mu+sigma*(pi/2)^(.5)
856 c***** and an adjustment to estimate the uncertainty in the mode.
857 c***** This adjustment was simply a standard error of the mean
858 c***** error of the mean = [Variance/(N-1)]^(.5)
859 c***** N was taken to be 16.
860
861     UPRLIM = XX(N50) + 1.81 * Q
862     LOWLIM = XX(N50) - 1.21 * Q

```

```

863      DO 30 J=IBOT,ITOP
864      IF (XX(J) .GT. UPR LIM) K = K - 1
865      IF (XX(J) .LT. LOWLIM) M = M + 1
866      30 CONTINUE
867      C*****
868      C***** NGOODP IS THE NUMBER OF XX RETAINED FOR MODAL CALCULATION
869      C*****
870      NGOODP=K-M+1
871      C*****
872      C***** SM IS THE AVERAGE VALUE OF X WITHIN (M,K) KEPT
873      C*****
874      DO 31 J=M,K
875      SM=SM+XX(J)
876      31 CONTINUE
877      SM=SM/NGOODP
878      C*****
879      C***** S IS THE VARIANCE (WHICH IS SCALED)
880      C*****
881      S=0.
882      DO 33 J=M,K
883      33 S=S+(XX(J)-SM)**2
884      S=2.*S/(NGOODP*0.86)
885      C*****
886      C***** XM - MU, SD - SIGMA, THE RAYLEIGH WIDTH PARAMETER (SCALED VAR)
887      C*****
888      XM=SM-SQRT(S*1.5708)
889      SD=SQRT(S)
890      C*****
891      C***** LIMIT THE RANGE OF THE DATA USING MU AND THE SD
892      C***** REDEFINE UPR LIM AND LOWLIM
893      C*****
894      LOWLIM= XM
895      UPR LIM = XM + 4.0 * SD
896      DO 34 J=M,K
897      IF (XX(J) .LT. XM) XX(J) = XM
898      IF (XX(J) .GT. UPR LIM) XX(J) = UPR LIM
899      34 CONTINUE
900      C*****
901      C***** RECALCULATE S USING THE FIXED UP DATA
902      C***** SET XMOD EQUAL TO MU + SIGMA
903      C*****
904      S=0.
905      DO 35 J=M,K
906      35 S=S+(XX(J)-XM)**2
907      S=S/(2.*NGOODP)
908      SD=SQRT(S)
909      XMOD=XM+SD
910      RETURN
911      C*****
912      C***** IF ENTIRE SAMPLE IS THROWN OUT USE XMOD = XX(M25)
913      C***** BASED ON EMPIRICAL EVIDENCE
914      C*****
915      40 IF(UCNT .LE. 54) GO TO 41
916      WRITE(KW,302)

```

```

917 302 FORMAT('1',11X,'Q',11X,'SEQ #',11X,'XMOD',10X,'XX(25)',9X,
918     @ 'XX(75)')
919     WRITE(KW,310)
920 310 FORMAT('+',5(5X,'_____'))/)
921     UCNT=0
922 41 XMOD = XX(N25)
923     WRITE(KW,305) Q,LBLREC,XMOD,XX(N25),XX(N75)
924 305 FORMAT(1X,F15.0,I15,3F15.4)
925     UCNT=UCNT+1
926     RETURN
927 C*****
928 C***** ALL COUNTS OUTSIDE THE BOUNDS, WRITE MESSAGE, SET PARMS.
929 C*****
930     45 CONTINUE
931         WRITE(KW,307) ngoodp,LBLREC,XMOD
932 307 FORMAT(' NGOODP =',i10,' AT SEQ # ',i10,' XMOD = ',F20.4)
933     RETURN
934 C*****
935 C***** NOT ENOUGH POINTS TO DO ANYTHING
936 C*****
937     50 WRITE(KW,308) NPTS
938 308 FORMAT(1H,'MISTAKE: SHOULDNT HAVE NPTS=',i4,'.LT.8')
939     RETURN
940     END
941 C*****
942 C*****
943 C*****
944     SUBROUTINE YRDAY(IYR,MNTH,IDAY,IYRDAY)
945 C*****
946 C***** COMPUTES YEAR DAYS ( EXCEPT ON CENTURIES)
947 C*****
948     DIMENSION ID(12)
949     DATA ID(1),ID(2),ID(3),ID(4),ID(5),ID(6),ID(7),ID(8),ID(9),ID(10),
950     @ID(11),ID(12)/1,32,60,91,121,152,182,213,244,274,305,335/
951     IYRDAY=(ID(MNTH)-1)+IDAY
952     IF((MOD(IYR,4).EQ.0).AND.(MNTH.GT.2)) IYRDAY=IYRDAY+1
953     RETURN
954     END
955 C*****
956 C*****
957 C*****
958     SUBROUTINE GMTYR (IYRDAY,IHOUR,MINUT,ISEC,GT,LOCAL,ITZOW)
959 C*****
960 C***** COMPUTES GREENWICH HOURS IN YEAR SINCE JAN 01 0000 Z
961 C***** IN DECIMAL FORM - DOUBLE PRECISION
962 C*****
963 C***** LOCAL = 0 TIME ALREADY IN GREENWICH
964 C***** = 1 LOCAL TIME ... MUST CONVERT TO GREENWICH BY ADDING ITZOW
965 C***** ITZOW IS POSITIVE FOR WEST LONGITUDE, NEGATIVE FOR EAST
966 C***** E.G. 4 IS NEAR BERMUDA
967 C*****
968     DOUBLE PRECISION GT
969     IF (LOCAL) 10,10,5
970     5 IHOUR = IHOUR + ITZOW

```

```

971 10 GT = DFLOAT((IYRDAY - 1) * 24 + I HOUR) + DFLOAT(MINUT)/60.
972 @+ DFLOAT(ISEC)/3600.
973 RETURN
974 END
975 subroutine binwindow(ibot,itop)
976 c***** added 7/15/89
977 c*****
978 c***** This routine is a FORTRAN variation of the pascal procedure
979 c***** "bins" listed in Real Time Processing with Inverted Echo Sounders
980 c***** by Robert Petrocelli.
981 c*****
982 c***** The coding has been simplified and adapted for use within the
983 c***** Memode code. A description of the variables used follows:
984 c*****
985 c***** xx array containing the travel time counts
986 c***** bin array of 96 bins representing 128 count interval from
987 c***** 0-8192*(1.5). The factor of 1.5 insures that BUNSFIXed
988 c***** files are suitably processed.
989 c***** max used to store the maximum number of occurrences for a bin
990 c***** kmax pointer to locate max in the array bin. Upon completion
991 c***** of the routine kmax will be used to specify a range
992 c***** within which the mode is most likely to be found.
993 c***** ibot index of the first travel time in xx to be used. ibot
994 c***** and itop are indices found by applying the PUNS window.
995 c***** Upon exit from the routine ibot and itop will contain the
996 c***** new indices for the useful travel times, xx(ibot) to
997 c***** xx(itop).
998 c***** itop index of the last travel time to be used by this routine
999 c***** toplim new upper travel time limit
1000 c***** botlim new lower travel time limit
1001 c*****
1002 c***** See Mr. Petrocelli's report for details. In short this routine
1003 c***** applies a window to refine the data before the ttmode
1004 c***** routine. The idea is that the true surface reflections will be most
1005 c***** probable returns and there is a time period within which all the true
1006 c***** echo would be expected to reside. The 13 bit range of 8192 is
1007 c***** divided into 64 intervals. The bin containing the largest number of
1008 c***** occurrences is also most likely to be the interval within which the
1009 c***** single travel time representative of the burst would be found.
1010 c***** To insure that all the true echos are encompassed the adjacent
1011 c***** intervals are included; all other bins are excluded from further
1012 c***** processing. The range of 3(128) counts is certainly capable of
1013 c***** enclosing the 200 count range that the main trace on a "healthy"
1014 c***** buns plot fits within.
1015 c*****
1016 integer*4 bin(100)
1017 integer toplim,botlim
1018 real*4 xx(100)
1019 common/commod/xx
1020 common/memocm/xmod,ngoodp,sa,apts
1021
1022 c*****
1023 c***** don't bother if only one point
1024 c*****

```

```

1025 if (ibot.eq.itop) return
1026 c*****
1027 c***** initialize max and array bin
1028 c*****
1029 max=0
1030 do 40 i=1,100
1031 bin(i)=0
1032 40 continue
1033 c*****
1034 c***** loop thru and increment the number occurrences within a particular
1035 c***** counts interval. Example: If ixx(i)=394 then k=4 and bin(4) is
1036 c***** increased by one. If ixx(i)=128 then k=1 and bin(1)=bin(1)+1.
1037 c***** Note that the first term of k is integer division.
1038 c*****
1039 do 10 i=ibot,itop
1040 k=(int(xx(i))-i)/128 +1
1041 bin(k)=bin(k)+1
1042 c*****
1043 c***** store the current maximum number of occurrences and its location
1044 c*****
1045 if (bin(k).gt.max) then
1046 max=bin(k)
1047 kmax=k
1048 endif
1049
1050 10 continue
1051 c*****
1052 c***** set the travel time limits
1053 c*****
1054
1055 toplim=(kmax+1)*128
1056 botlim=(kmax-2)*128
1057 c*****
1058 c***** use these limits to find indices of fist and last points in the
1059 c***** new window.
1060 c*****
1061 c***** clamp down to find the bottom of the window.
1062 c*****
1063 do 20 i=ibot,itop
1064 if (xx(i).ge.botlim) then
1065 ibot=i
1066 goto 100
1067 endif
1068 20 continue
1069 c*****
1070 c***** do the similar operation to find the top of the window.
1071 c*****
1072
1073 100 do 30 i=itop,ibot,-1
1074 if (xx(i).le.toplim) then
1075 itop=i
1076 return
1077 endif
1078 30 continue

```

1079 end

3.5 DETIDE_AUG90.FOR

```

1 c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 c%%%%%%%% detide_aug90.for
3 c%%%%%%%%
4 c%%%%%%%% revised from detide_jul88.for to deal with *.fill files from
5 c%%%%%%%% fill_aug90.for. These differ from detide_jul88.for in that they
6 c%%%%%%%% contain only travel time and yearhour ( no tmp, prs, or am col-
7 c%%%%%%%% umns).
8 c%%%%%%%%
9 c%%%%%%%% Fields 4-aug-90
10 C*****
11 C*****                DETIDE.JULY88.FOR
12 C*****
13 C***** PROGRAM TO DETIDE THE MEDIAN/MODE VALUES OF TRAVEL TIMES.
14 C***** DETERMINES THE TIDAL HEIGHT IN CM AND CONVERTS IT TO SECONDS.
15 C***** IT IS ASSUMED THAT THE DATASET HAS BEEN PROCESSED THROUGH THE
16 C***** FILL PROGRAM, SO THAT THERE ARE NO GAPS. THE OUTPUT DATASET
17 C***** CONTAINS, THE MEASURED TAU'S, THE DETIDED TAU'S AND THE TIDE,
18 C***** AS WELL AS THE OTHER INPUT PARAMETERS.
19 C*****
20 C***** FORTRAN UNIT NUMBERS DESIGNATED AS FOLLOWS:
21 C*****      KR (UNIT 15) CONTROL CARD INPUT FILE
22 C*****      KR1 (UNIT 18) INPUT DATASET FROM FILL
23 C*****      KW (UNIT 11) OUTPUT USERS LOG
24 C*****      KW1 (UNIT 17) OUTPUT DATA FILE
25 C*****
26      INTEGER*4 YEAR,TOTREC,HUNDRD,RECOUT
27      CHARACTER*60 HEADR
28      CHARACTER*10 LOCM
29      REAL*4 TIME(200),DELT
30      REAL*4 TTB(200),ZO,MTIDCM,TTID(200)
31      REAL*4 XODTID(200)
32      REAL*8 DTIME
33 C*****
34      COMMON/TIDCON/ H(8), PHI(8),VU(8),FNODE(8)
35      COMMON/TPARMS/ZO,IYR,ICALVU
36 C*****
37      PARAMETER (HUNDRD=200)
38      PARAMETER (KR=15,KW=11,KW1=17,KR1=18)
39      NAMELIST/CARD1/HEADR
40      NAMELIST/CARD2/NFIRST,NLAST,IYR,DELT
41      NAMELIST/CARD3/H
42      NAMELIST/CARD4/PHI
43      NAMELIST/CARD5/FNODE
44      NAMELIST/CARD6/VU
45      NAMELIST/CARD7/CBAR,PFACTR,LOCM
46      DATA ICALVU/0/
47      DATA CBAR/1510./,PFACTR/1.0/,LOCM/'N.U.'/
48      DATA MTIDCM/0.0/,TTID/200*0.0/
49      DATA ZO/0.0/
50      DATA H/8*0.0/,PHI/8*0.0/,FNODE/8*1.0/,VU/8*0.0/
51      DATA RECOUT/0/,IPRNT/0/
52 C*****

```

```

53 C***** OPEN I/O UNITS AND FILES
54 C*****
55     OPEN(UNIT=KR,STATUS='OLD',FORM='FORMATTED',READONLY)
56     OPEN(UNIT=KR1,STATUS='OLD',FORM='FORMATTED',READONLY)
57     OPEN(UNIT=KW,STATUS='NEW',FORM='FORMATTED')
58     OPEN(UNIT=KW1,STATUS='NEW',FORM='FORMATTED')
59 C*****
60 C*****  READ CONTROL PARAMETERS
61 C*****  NFIRST:  THE REC # OF THE FIRST 'GOOD' RECORD
62 C*****  NLAST:  LAST REC # TO BE PROCESSED
63 C*****
64     READ(KR,NML=CARD1)
65     READ(KR,NML=CARD2)
66     READ(KR,NML=CARD3)
67     READ(KR,NML=CARD4)
68     READ(KR,NML=CARD5)
69     READ(KR,NML=CARD6)
70     READ(KR,NML=CARD7)
71 C*****
72     WRITE(*,42)
73 42  FORMAT(1X,/,5X,' DETIDE IS NOW PROCESSING YOUR DATA!!'//)
74 C*****
75 C*****  PRINT CONTROL PARAMETERS
76 C*****
77     WRITE(KW,400)
78 400 FORMAT(1H1,T50,'* * * 'PROGRAM DETIDE * * *',//)
79     WRITE(KW,405) HEADR
80 405 FORMAT(/T35,A60)
81     WRITE(KW,410) NFIRST,NLAST,CBAR,PFACTR,LOCN
82 410 FORMAT(/T15,'NFIRST',T26,'NLASt',T37,
83     @'CBAR',T45,'PFACTR',T57,'LOCN',
84     @/,',+',T11,5(2X,'-----'),
85     @//,T10,2I10,2F10.2,6X,A4)
86     WRITE(KW,415) IYR,DELT
87 415 FORMAT(/8X,'IYR =',I10//8X,'DELT=',F12.7)
88     WRITE(KW,420)H,PHI,VU,FNODE
89 420 FORMAT(/T10,'TIDAL PARAMETERS ',
90     @//T17,'M2',T27,'N2',T37,'S2',T47,'K2',T57,
91     @'K1',T67,'Q1',T77,'P1',T87,'Q1',
92     @//T9,'H',8F10.2,
93     @/T7,'PHI',8F10.2,
94     @/T6,'VO+U',8F10.3,
95     @/T9,'F',8F10.3//)
96     WRITE(KW,425)
97 425 FORMAT(/T40,'FIRST RECORD OF EACH BLOCK PROCESSED',
98     @//T16,'TTB',T26,'XODTID',T37,'TTID',T48,'TIME',
99     @/,',+',T12,2X,'-----',3(3X,'-----'),//)
100 C*****
101 C*****  COMPUTE TRAVEL TIME CONVERSION FACTOR, TTCONV CONVERTS TIDE
102 C*****  FROM CM TO SEC.  CBAR IN (M/SEC) IS AVERAGE SOUND VELOCITY
103 C*****  FROM MATTHEWS TABLES FOR THE LOCATION AND DEPTH OF THE IES.
104 C*****  PFACTR = 1 + DEPTH * 1.1E-5,  WHERE 1.1E-5=(1/C)(DC/DP)
105 C*****  DEFAULT FOR PFACTR IS 1.0
106 C*****

```



```

107      TTCOVV = 2.0E-2 / (CBAR * PFACTR)
108 C*****
109 C***** READ DATA IN BLOCKS OF 200 TO IMPROVE I/O EFFICIENCY.
110 C*****
111 C*****
112 C***** COMPUTE TOTAL # OF RECORDS TO BE READ, AND THE
113 C***** CORRESPONDING NUMBER OF BLOCKS, PLUS A REMAINDER.
114 C*****
115      25 CONTINUE
116      TOTREC = NLAST-NFIRST + 1
117      NBLK = TOTREC/HUNDRD
118      NGET = MOD( TOTREC , HUNDRD )
119      IF( NBLK .LT. 1) NGET = TOTREC
120      NGO = HUNDRD
121      NRDBLK = 0
122 C*****
123 C***** READ BLOCK OF DATA, CHECKING THAT THERE ARE AT LEAST 200
124 C*****
125      30 IF(NRDBLK .GE. NBLK) NGO = NGET
126      IF(NGO .EQ. 0) GO TO 55
127      READ(KR1,430,END=999) (TTB(I),TIME(I),I=1,NGO)
128      430 FORMAT(2E15.7)
129      NRDBLK = NRDBLK + 1
130 C*****
131 C***** PROCESS ONE BLOCK OF DATA AT A TIME
132 C*****
133      DO 45 J=1,NGO
134 C*****
135 C***** CONVERT TIME TO DOUBLE PRECISION
136 C*****
137      40 CONTINUE
138      DTIME = DBLE(TIME(J))
139 C*****
140 C***** NOW COMPUTE TIDAL HEIGHT
141 C*****
142      CALL TIDE(DTIME,MTIDCM)
143 C*****
144 C***** ALL TRAVEL TIMES ARE IN SECONDS, SO SCALE TIDE TO SECONDS AND
145 C***** SUBTRACT FROM THE INPUT DATA
146 C*****
147      TTID(J) = (TTCOVV * MTIDCM)
148      XODTID(J) = TTB(J) - TTID(J)
149      45 CONTINUE
150 C*****
151 C***** PRINT FIRST RECORD OF EACH BLOCK TO OUTPUT LOG
152 C*****
153      WRITE (KW,435) TTB(1),XODTID(1),TTID(1),TIME(1)
154      435 FORMAT(/T10,F11.5,1X,2F11.5,F10.2)
155      IPRNT = IPRNT + 1
156 C*****
157 C***** DUMP IT ALL ON THE DISK OUTPUT
158 C*****
159      WRITE(KW1,440) (TTB(J),XODTID(J),TTID(J),TIME(J),J=1,NGO)
160      440 FORMAT(4E15.7)

```

```

161     RECOUT = RECOUT + NGO
162     50 IF(NGET .LT. NGO) GO TO 30
163 C*****
164 C*****     END OF PROCESSING - WRAP UP
165 C*****
166     55 CONTINUE
167     WRITE(KW,450) RECOUT
168     450 FORMAT(/T10,' *** PROCESSING ENDED AT SPECIFIED RECORD--',
169     @I6,' RECORDS WRITTEN ON UNIT 12'//)
170     WRITE(*,450) RECOUT
171     WRITE(*,43)
172     STOP
173     999 WRITE(KW,455) RECOUT,I
174     455 FORMAT(/T10,'>>>> ERROR: UNEXPECTED END OF DATA <<<<',
175     @/T10,I6,' RECORDS PROCESSED',I6,' ADDITIONAL RECORDS',
176     @' READ BEFORE EOF'//)
177 C*****
178     WRITE(*,455) RECOUT,I
179     WRITE(*,43)
180     43  FORMAT(6X,' FINISHED!!'//)
181     STOP
182     END
183 C*****
184 C*****
185 C*****
186     SUBROUTINE TIDE(GYRHR,ZT)
187 C*****
188 C*****     TIDAL COMPONENTS ORDERED: M2, M2, S2, K2, K1, O1, P1, Q1
189 C*****     REQUIRES HALF AMPLITUDES: H(L), CM
190 C*****     PHASES: PHI(L) DEGREES, GREENWICH EPOCH
191 C*****     GYRHR = GREENWICH HOURS SINCE THE BEGINNING OF IYEAR
192 C*****     VOANDU IS THE ARGUMENT OF EQUILIBRIUM TIDE (GEOPOTENTIAL)
193 C*****     AT THE STARTING TIME AT GREENWICH MERIDIAN (VO + U
194 C*****     IN TABLE 15 OF COAST AND GEODETIC SURVEY SPECIAL PUB. 98).
195 C*****
196 C*****
197 C*****     METHOD:  ZT = ZO + SUM (H(L) * COS(ARG(L))
198 C*****     WHERE:
199 C*****     ARG(L) = TPI * (FREQ(L) * TIME + (VU(L) - PHI(L)) / 360.0)
200 C*****
201 C*****     VU(L) IS THE ARGUMENT (DEGREES) OF EQUILIBRIUM TIDAL
202 C*****     CONSTITUENT L AT REFERENCE TIME = NDAYS
203 C*****
204 C*****     ZT IS THE CALCULATED TIDE
205 C*****
206 C*****     NOTE:  GIVES CAREFUL ATTENTION TO USING DOUBLE PRECISION
207 C*****     ONLY WHERE NECESSARY
208 C*****
209     COMMON/TIDCON/ H(8),PHI(8),VU(8),FNODE(8)
210     COMMON/TPARMS/ZO,IYEAR,ICALVU
211     REAL*4 VOANDU(8),PCYCLS(8)
212     INTEGER*4 CALLCT
213     DOUBLE PRECISION FREQ(8),GYRHR,DAYS,FLOAT,DCYCLS
214     PARAMETER (TPI=6.2831853,KW=1)

```

```

215     DATA CALLCT/0/
216     DATA FREQ(1)/1.9322736D0/,
217     @   FREQ(2)/1.8959820D0/,
218     @   FREQ(3)/2.0D0/,
219     @   FREQ(4)/2.0054758D0/,
220     @   FREQ(5)/1.0027379D0/,
221     @   FREQ(6)/0.9295357D0/,
222     @   FREQ(7)/0.9972621D0/,
223     @   FREQ(8)/0.8932441D0/
224     DATA VOANDU/8.3,65.7,0.0,217.5,18.9,345.2,349.8,42.6/
225     C*****
226     C***** COUNT NUMBER TIMES THIS SUBROUTINE HAS BEEN CALLED.
227     C***** CHECK TO SEE IF VU MUST BE CALUCLATED (ICALVU=1)
228     C***** OR IF IT WAS SUPPLIED (ICALVU=0).
229     C*****
230     CALLCT=CALLCT+1
231     IF (ICALVU .EQ. 0) GO TO 25
232     C*****
233     C***** VU MUST BE CALCULATED:
234     C***** CALCULATE # OF DAYS FROM 1 JAN 1900 TO 1 JAN IYEAR
235     C***** LEAP DAYS ON YEARS DIVISIBLE BY 4 EXCEPT 1900
236     C***** (CENTURY NOT LEAP YR UNLESS DIVISIBLE BY 400).
237     C***** CALCULATE # OF CYCLES PER DAY AT EACH FREQUENCY,
238     C***** USE ONLY FRACTIONAL NUMBER OF CYCLES.
239     C*****
240     10 LPDYS = (IYEAR - 1900) / 4
241     NDAYS = (IYEAR - 1900) * 365 + LPDYS
242     DO 15 L = 1, 8
243     DCYCLS = FREQ(L) * NDAYS
244     DCYCLS = DCYCLS - FLOAT(IDINT(DCYCLS))
245     VU(L) = SNGL(DCYCLS) * 360.0 + VOANDU(L)
246     IF (VU(L) .GE. 360.) VU(L) = VU(L) - 360.
247     15 CONTINUE
248     ICALVU = 0
249     WRITE(KW,400) IYEAR,(VU(L),L=1,8)
250     400 FORMAT(//1H0,'EQUILIBRIUM TIDE ARGUMENTS (DEGREES) ',
251     @' AT THE BEGINNING OF ',I4,8F8.1//1H0)
252     C*****
253     C***** VU ALREADY CALCULATED:
254     C***** IF FIRST CALL TO SUBROUTINE, CALCULATE THE DIFFERENCES
255     C***** BETWEEN THE PHASES VU AND PHI. THEN CONVERT TO CYCLES/DAY.
256     C*****
257     25 CONTINUE
258     IF (CALLCT .GT. 1) GO TO 35
259     DO 30 L=1,8
260     PCYCLS(L) = (VU(L) - PHI(L)) / 360.0
261     30 CONTINUE
262     C*****
263     C***** MAIN LOOP FOR CALCULATING THE TIDE AT TIME=GYRHR
264     C*****
265     35 CONTINUE
266     DAYS = GYRHR / 24.0D0
267     ZT = Z0
268     DO 40 L=1, 8

```

```
'9      DCYCLS = FREQ(L) * DAYS
'0      DCYCLS = DCYCLS - FLOAT(IDINT(DCYCLS))
'1      ARG1 = (SMGL(DCYCLS) + PCYCLS(L)) * TPI
'2      ZT = ZT + FNODE(L) * H(L) * COS(ARG1)
'3      40 CONTINUE
'4      RETURN
'5      END
```

3.6 DESPIKE_AUG90.FOR

```

1 c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7 c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8 C*****
9 C*****          DESPIKE.JULY88.FOR
10 C*****
11 C*****  PROGRAM TO DESPIKE THE DETIDED TRAVEL TIMES.  SPIKES ARE
12 C*****  IDENTIFIED IN TWO WAYS:  1) THE DATA MUST BE WITHIN A SPECIFIED
13 C*****  RANGE OF GOOD TRAVEL TIMES; AND 2) THE CHANGE BETWEEN TWO
14 C*****  ADJACENT MEASUREMENTS MUST BE LESS THAN A SPECIFIED RATE.  IF
15 C*****  EITHER OF THESE TWO CRITERIA IS NOT MET, THE DATA POINT IS
16 C*****  REPLACED WITH AN INTERPOLATED VALUE.
17 C*****
18 C*****  PROGRAM ORIGINALLY WRITTEN BY BY J. GUNN 1978,
19 C*****  BUT HAS BEEN REWRITTEN SEVERAL TIMES SINCE 1981 BY K. TRACEY.
20 C*****  CONVERTED TO VAX BY SLW
21 C*****
22 C*****  NOTE, THE SAV... ARRAYS MUST BE DIMENSIONED THE SAME AS OR
23 C*****  GREATER THAN MAXBAD.
24 C*****
25 C*****  I/O UNITS:
26 C*****      KR (UNIT 17) CONTROL CARD FILE
27 C*****      KR1 (UNIT 19) INPUT DATASET FROM DETIDE
28 C*****      KW (UNIT 11) OUTPUT USERS LOG
29 C*****      KW1 (UNIT 18) OUTPUT DATASET
30 C*****      KW2 (UNIT 16) OUTPUT LISTINGS FILE
31 C*****
32      INTEGER*4 IRECCT, BADREC, IBADCT, NBAD
33      INTEGER*4 FLAGCT
34      LOGICAL FLAG
35      CHARACTER*80 HEADR
36      INTEGER*4 MAXBAD
37      REAL*4 KNT1
38      REAL*4 SAVX0(100), SAVXOD(100), INBTWN
39      REAL*4 X0, TIME, TIDE
40      REAL*4 XOFIL, XODFIL
41      REAL*4 SAVTID(100), SAVTIM(100), KNT1S
42      REAL*8 X1SAV(100), XODIF, XOMAX
43      REAL*8 RMAVG1, SMAX1
44      PARAMETER (KR1=19, KW=11, KW1=18, KR=17, KW2=16)
45      PARAMETER (ON='<===', OFF=' ')
46      NAMELIST/CARD1/HEADR
47      NAMELIST/CARD2/TINTVL, VMAX, VMIN
48      NAMELIST/CARD3/SLOPE1, RMAVG1, LAVG1
49      DATA IRECCT/0/, LINECT/55/
50      DATA IBADCT/0/, NBAD/0/, FLAGCT/0/, MAXBAD/100/
51 C*****
52 C***** OPEN I/O UNITS AND FILES

```

```

53 C*****
54     OPEN(UNIT=KR,STATUS='OLD',FORM='FORMATTED',READONLY)
55     OPEN(UNIT=KR1,STATUS='OLD',FORM='FORMATTED',READONLY)
56     OPEN(UNIT=KW,STATUS='NEW',FORM='FORMATTED')
57     OPEN(UNIT=KW1,STATUS='NEW',FORM='FORMATTED')
58     OPEN(UNIT=KW2,STATUS='NEW',FORM='FORMATTED')
59 C*****
60 C*****  READ AND WRITE IN THE CONTROL CARDS
61 C*****
62     READ(KR,NML=CARD1)
63     READ(KR,NML=CARD2)
64     READ(KR,NML=CARD3)
65     write(*,42)
66 42  format(1x,/,5x,'DESPIKING your data set---> Pondering ',/)
67     WRITE(KW,608) HEADR
68 608  FORMAT(/,45X,'..... DESPIKE .....',
69      @/'0',A80)
70     WRITE(KW,610) TINTVL, VMAX, VMIN,
71      @SLOPE1, RMAVG1, LAVG1
72 610  FORMAT(/,5X,'CONTROL CARDS',
73      @T25,9X,'TINTVL',11X,'VMAX',11X,'VMIN'/
74      @T25,3(5X,F10.4)//
75      @T25,9X,'SLOPE1',9X,'RMAVG1',10X,'LAVG1'/
76      @T25,2(5X,F10.4),5X,I10//)
77 C*****
78 C*****  INITIALIZE VARIABLES
79 C*****  KNT1 AND KNT1S ARE INDEXES OF THE X1SAV ARRAY
80 C*****  LAST1 IS PREVIOUS VALUE OF KNT1
81 C*****  RLAVG1 IS NUMBER OF POINTS IN RUNNING AVERAGE
82 C*****  SMAX1 IS MAXIMUM ALLOWED CHANGE IN SECONDS PER SAMPLING PERIOD
83 C*****
84     KNT1 = (LAVG1 + 1.0)/2.0
85     KNT1S = KNT1
86     LAST1 = LAVG1
87     RLAVG1 = LAVG1
88     SMAX1 = SLOPE1*TINTVL
89 C*****
90 C*****  IF THE INITIAL RUNNING AVERAGES WERE NOT SUPPLIED,
91 C*****  COMPUTE THEM FROM THE FIRST LAVG1 POINTS IN THE DATA SET
92 C*****  NOTE THAT THESE POINTS WILL NOT BE DESPIKED.
93 C*****
94     IF (RMAVG1 .NE. 0.0) GO TO 30
95     DO 20 I=1,LAVG1
96     READ(KR1,400,END=900)X0,XODTID,TIDE,TIME
97     IRECCT = IRECCT + 1
98     WRITE(KW1,400) X0,XODTID,TIDE,TIME
99     X1SAV(I) = XODTID / RLAVG1
100    RMAVG1 = RMAVG1 + X1SAV(I)
101    20 CONTINUE
102    GO TO 100
103 C*****
104 C*****  IF INITIAL RUNNING AVERAGES WERE SUPPLIED,
105 C*****  INITIALIZE THE PREVIOUS POINTS ARRAY TO THE RUNNING AVERAGES
106 C*****

```

```

107     30 CONTINUE
108         DO 40 I=1,LAvg1
109             X1SAV(I) = RNAVG1 / RLAVG1
110     40 CONTINUE
111 C*****
112 C*****          MAIN PROCESSING LOOP
113 C*****  REMOVE THE SPIKES FROM THE DATA
114 C*****
115     100 CONTINUE
116         READ (KR1,400,END=900) X0,XODTID,TIDE,TIME
117     400 FORMAT(4E15.7)
118         IRECCT = IRECCT + 1
119 C*****
120 C*****  TEST EACH DETIDED POINT
121 C*****  SEE IF IT IS OUTSIDE THE LIMITS BEFORE CHECKING THE SLOPE
122 C*****
123         IF (XODTID .GE. VMAX) GO TO 105
124         IF (XODTID .LE. VMIN) GO TO 105
125         XODIF = XODTID - RNAVG1
126         XOMAX = KWT1S * SMAX1
127         IF (DABS(XODIF) .GT. XOMAX) GO TO 105
128         GO TO 110
129 C*****
130 C*****  FOR BAD POINTS, OPEN UP THE WINDOW AND SAVE THE VALUES.
131 C*****  TO BE WRITTEN OUT LATER WITH THE INTERPOLATED VALUES.
132 C*****
133     105 CONTINUE
134         KWT1S = KWT1S + 1.0
135         NBAD = NBAD + 1
136         IF (NBAD .EQ. 1) BADREC = IRECCT
137         IF (NBAD .GT. MAXBAD) GO TO 2000
138         SAVX0(NBAD) = X0
139         SAVXOD(NBAD) = XODTID
140         SAVTID(NBAD) = TIDE
141         SAVTIM(NBAD) = TIME
142         GO TO 100
143 C*****
144 C*****  IF THE POINT WAS O.K., THEN
145 C*****  RESET THE WINDOW, ACCOUNTING FOR LENGTHENING IF FILLED
146 C*****
147     110 CONTINUE
148         KWT1S = KWT1 + NBAD/2.0
149 C*****
150 C*****  IF PREVIOUS POINT(S) WERE GOOD - SKIP THIS LOOP
151 C*****  IF NECESSARY, WRITE A NEW HEADER FOR THE LOG PAGE
152 C*****
153         IF (NBAD .EQ. 0) GO TO 115
154         IF(LINECT .LE. 54) GO TO 77
155         WRITE(KW2,72)
156     72 FORMAT('1',7X,'RNAVG',8X,'NBAD',8X,'REC#',8X,'XOLD',7X,
157     @'XOFIL',4X,'XDTIDOLD',4X,'XDTIDFIL',8X,'TTID',10X,'TIME',
158     @/'+',9('-----'),'__')
159         LINECT = 0
160     77 WRITE(KW2,632) RNAVG1, NBAD

```

```

161      632 FORMAT(/3X,F10.6,2X,I10)
162      PRVXOD = X1SAV(LAST1) * RLAVG1
163      DLTXOD = XODTID - PRVXOD
164      XODINC = DLTXOD / (NBAD + 1)
165 C*****
166 C*****          INTERPOLATING LOOP
167 C***** FILL IN THE BAD POINTS WITH INTERPOLATED POINTS
168 C*****
169      DO 114 I = 1, NBAD
170 C*****
171 C***** IF VALUE OF BAD POINT IS BETWEEN TWO GOOD ONES - SET FLAG
172 C*****
173      FLAG = OFF
174      INBTWN = (XODTID - SAVXOD(I)) * (SAVXOD(I) - PRVXOD)
175      IF(INBTWN .LE. 0.0) GO TO 112
176      113 FLAG = ON
177      FLAGCT = FLAGCT + 1
178 C*****
179 C*****      ....UPDATE THE FIFO RUNAVG STACK WITH FLAGGED POINTS .....
180 C*****
181      LAST1 = LAST1 + 1
182      IF (LAST1 .GT. LAVG1) LAST1 = 1
183      RNAVG1 = RNAVG1 - X1SAV(LAST1)
184      XNOW = DBLE(SAVXOD(I)) / RLAVG1
185      RNAVG1 = RNAVG1 + XNOW
186      X1SAV(LAST1) = XNOW
187 C*****
188 C*****      ....END OF UPDATING FIFO STACK WITH INTERPOLATED DATA....
189 C*****
190      112 CONTINUE
191      XODFIL = PRVXOD + XODINC * I
192      XOFIL = XODFIL + SAVTID(I)
193 C*****
194 C***** WRITE INTERPOLATED POINTS TO OUTPUT LOG AND DATASET
195 C*****
196      IF(I .GT. 1) GO TO 78
197      WRITE(KW2,634) BADREC, SAVXO(I), XOFIL,
198      @SAVXOD(I), XODFIL, SAVTID(I), SAVTIM(I),FLAG
199      634 FORMAT(' ',26X,I10,5(2X,F10.6),2X,F12.5,2X,A4)
200      LINECT = LINECT + 2
201      GO TO 79
202      78 WRITE(KW2,635) BADREC, SAVXO(I), XOFIL,
203      @SAVXOD(I), XODFIL, SAVTID(I), SAVTIM(I),FLAG
204      635 FORMAT(' ',26X,I10,5(2X,F10.6),2X,F12.5,2X,A4)
205      LINECT = LINECT + 1
206      79 WRITE(KW1,400) XOFIL, XODFIL, SAVTID(I), SAVTIM(I)
207      BADREC = BADREC + 1
208      114 CONTINUE
209 C*****
210 C***** INCREMENT THE COUNTERS
211 C*****
212      IBADCT = IBADCT + NBAD
213      NBAD = 0
214      115 CONTINUE

```



```

215 C*****
216 C***** SAVE THE LAST LNGAVG POINTS IN A FIFO STACK
217 C***** AND UPDATE THE RUNNING AVERAGES
218 C***** NOTE: THE STACK CONTAINS THE VALUE OF EACH POINT DIVIDED BY
219 C***** THE NUMBER OF POINTS IN THE RUNNING AVERAGE
220 C***** THIS WAY WE ONLY HAVE TO DIVIDE ONCE PER POINT
221 C*****
222     LAST1 = LAST1 + 1
223     IF (LAST1 .GT. LAVG1) LAST1 = 1
224     RNAVG1 = RNAVG1 - X1SAV(LAST1)
225     X1SAV(LAST1) = DBLE(XODTID) / RLAVG1
226     RNAVG1 = RNAVG1 + X1SAV(LAST1)
227 C*****
228 C***** WRITE THE GOOD POINTS TO THE OUTPUT DATA SET
229 C***** THEN CONTINUE TH PROCESSING LOOP
230 C*****
231     WRITE(KW1,400) XO,XODTID,TIDE,TIME
232     GO TO 100
233 C*****
234 C***** IF TOO MANY CONSECUTIVE BAD RECORDS -
235 C***** PRINT MESSAGE AND STOP
236 C*****
237     2000 CONTINUE
238     WRITE(KW,680) MAXBAD
239     680 FORMAT(//5X,'MORE THAN',I4,' CONSECUTIVE BAD POINTS FOUND',
240     @//5X,'***** RUN TERMINATED *****')
241     WRITE(*,680) MAXBAD
242     STOP 999
243 C*****
244 C***** WRAP UP - WRITE INFO TO OUTPUT LOG
245 C*****
246     900 CONTINUE
247     WRITE(KW,690) IRECCT, IBADCT, FLAGCT
248     690 FORMAT(//5X,I5,' RECORDS WERE PROCESSED',
249     @//5X,I5,' BAD POINTS WERE REPLACED',
250     @//5X,I5,' REPLACEMENTS WERE FLAGGED')
251     write(*,43)
252     43 format(1x,/,5x,'Pondering Finished!')
253     STOP
254     END

```

3.7 SEACOR_AUG90.FOR

```

1  c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5  c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6  c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7  c%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8  c
9  c"23-Nov-1990 revised to include the new seasonal
10 c" corrections for my Stephen, Karen, and DRW. The
11 c" correction factors used are location specific.
12 c" There are three regions. The SYNOP data falls into two
13 c" of the regions; The inlet array is region 1, and the
14 c" central array is in region 2.
15 c"
16 c" The control file now requires a third namelist containing
17 c" the numerical specification of the region (1 or 2).
18 c"
19 C*****
20 C*****          SEACOR.JULY88.FOR
21 C*****
22 C*****  A SEASONAL CORRECTION FACTOR IS ADDED TO THE MEASURED AND
23 C*****  DETIDED TRAVEL TIMES.  THE CORRECTION FACTOR FOR EACH
24 C*****  SAMPLING PERIOD IS CALCULATED BY A LINEAR INTERPOLATION
25 C*****  BETWEEN THE ARRAY OF MONTHLY CORRECTION FACTORS (CF).
26 C*****
27 C*****  FOR THIS VERSION, THE VALUES IN ARRAY CF ARE FOR THE GULF
28 C*****  STREAM REGION.  THEY WERE CALCULATED USING ISELIN'S (1940)
29 C*****  HYDROGRAPHIC DATA.
30 C*****
31 C*****  I/O UNITS:
32 C*****      KRCTRL (UNIT 1) - CONTROL FILE
33 C*****      KWLOG  (UNIT 2) - OUTPUT USER'S LOG
34 C*****      KRIES  (UNIT 3) - IES INPUT DATA FILE
35 C*****      KWIES  (UNIT 4) - OUTPUT FILE OF SEASONALLY CORRECTED
36 C*****      IES DATA
37 C*****
38 CHARACTER*2 YES/'YE'/,FRSTYR/'NO'/,SCNDYR/'NO'/
39 CHARACTER*80 HEADR
40 INTEGER*4 HI, LOW
41 integer region
42 REAL*4 TT(200),TTDTID(200),TID(200),TIME(200)
43 c REAL*4 TMP(200),PRS(200),AMB(200)
44 REAL*4 TTSCF(200),TDTSCF(200)
45 REAL*4 MNTH(24),CF(24),CF1(24),CF2(24),CF3(24),DSLOPE(24)
46 REAL*4 CFDIF,MDIF,YREND(2),YRHR
47 REAL*8 DEL,LSTDEL,TDIF,SCF
48 REAL*8 DYRHR, DHMNTH, DLMNTH
49 PARAMETER(KRCTRL=1,KWLOG=2,KRIES=3,KWIES=4)
50 NAMELIST/CARD1/HEADR
51 NAMELIST/CARD2/WPTS,NOYRS,FRSTYR,SCNDYR
52 namelist/card3/region

```

```

53      DATA WTOTL/0/,YREND/8760.0,8760.0/
54 C*****
55 C*****  INITIALIZE THE YEARHOUR AND CORRECTION FACTOR ARRAYS
56 C*****  PROGRAM ASSUMES THAT THE YEARHOURS ARE FOR THE FIRST
57 C*****  DAY OF EACH MONTH.
58 C*****  THESE VALUES ARE FOR THE GULF STREAM REGION.
59 C*****
60      DATA MNTH/0., 744., 1416., 2160., 2880., 3624.,
61      @      4344., 5088., 5832., 6552., 7296., 8016.,
62      @      0., 744., 1416., 2160., 2880., 3624.,
63      @      4344., 5088., 5832., 6552., 7296., 8016./
64      DATA CF1/0.000551553, 0.000143760, 0.0, 0.000244602,
65      /      0.000581494, 0.000730828, 0.000743656, 0.000727823,
66      /      0.000746362, 0.000850607, 0.000960953,0.000886401,
67      /      0.000551553, 0.000143760, 0.0, 0.000244602,
68      /      0.000581494, 0.000730828, 0.000743656, 0.000727823,
69      /      0.000746362, 0.000850607, 0.000960953,0.000886401/
70      DATA CF2/0.00069029, 0.00028889, 0.0, 0.00001659,
71      /      0.00034654, 0.00080489, 0.00115238, 0.00134841,
72      /      0.00143492, 0.00135541, 0.00116881, 0.00097876,
73      /      0.00069029, 0.00028889, 0.0, 0.00001659,
74      /      0.00034654, 0.00080489, 0.00115238, 0.00134841,
75      /      0.00143492, 0.00135541, 0.00116881, 0.00097876/
76      DATA CF3/0.00038513, 0.00012636, 0.0, 0.00006714,
77      /      0.00046289, 0.00107709, 0.00155502, 0.00174333,
78      /      0.00178230, 0.00169593, 0.00135379, 0.00082930,
79      /      0.00038513, 0.00012636, 0.0, 0.00006714,
80      /      0.00046289, 0.00107709, 0.00155502, 0.00174333,
81      /      0.00178230, 0.00169593, 0.00135379, 0.00082930/
82 C*****
83 C*****  OPEN THE INPUT AND OUTPUT DATASETS
84 C*****
85      OPEN(UNIT=KRIES,STATUS='OLD',FORM='FORMATTED',READONLY)
86      OPEN(UNIT=KWIES,STATUS='NEW',FORM='FORMATTED')
87 C*****
88 C*****  READ CONTROL CARD FILE
89 C*****
90      READ(KRCTRL,NML=CARD1)
91      READ(KRCTRL,NML=CARD2)
92      READ(KRCTRL,NML=CARD3)
93  if (region.eq.1) then
94  do 67 i=1,24
95  cf(i)=cf1(i)
96  67 continue
97  else if (region.eq.3) then
98  do 68 i=1,24
99  cf(i)=cf3(i)
100  68 continue
101  else
102  do 69 i=1,24
103  cf(i)=cf1(i)
104  69 continue
105  endif
106

```

```

107     WRITE(*,42)
108  42  FORMAT(1X,/,5X,' SEACOR HAS TAKEN OVER AND IS PROCESSING.',/)
109  C*****
110  C*****  IF A LEAP YEAR, ADJUST THE BEGINNING TIMES OF EACH MONTH
111  C*****
112     IF (FRSTYR .EQ. YES) THEN
113         YREND(1)=8784.0
114         DO 100 LP=3,12
115             MNTH(LP)=MNTH(LP)+24.0
116  100  CONTINUE
117     ELSE IF (SCNDYR .EQ. YES) THEN
118         YREND(2)=8784.0
119         DO 110 LP=15,24
120             MNTH(LP)=MNTH(LP)+24.0
121  110  CONTINUE
122     END IF
123  C*****
124  C*****  IF DATASET SPANS TWO YEARS, THE FIRST YEAR WILL HAVE
125  C*****  NEGATIVE YEARHOURS.  SO RESET THE BEGINNING MONTHS
126  C*****
127     IF (NOYRS .EQ. 2) THEN
128         DO 115 LP=1,12
129             MNTH(LP)=MNTH(LP)-YREND(1)
130  115  CONTINUE
131     END IF
132  C*****
133  C*****  WRITE HEADER INFO TO USER'S LOG
134  C*****
135     20 WRITE(KWLOG,405) HEADR
136  405 FORMAT('1',T35,'***** SEASONAL CORRECTION FACTOR PROGRAM ',
137     @ '*****'//'0',A80)
138     WRITE(KWLOG,410) NPTS,NOYRS, FRSTYR, SCNDYR
139  410 FORMAT('0',I10,' SAMPLING PERIODS ARE TO BE READ',
140     @ '/'0 THE DATASET SPANS',I2,' YEAR(S)',
141     @ '/' IS THE FIRST YEAR A LEAP YEAR? ',A4,
142     @ '/' IS THE SECONDD YEAR A LEAP YEAR?',A4)
143     WRITE(KWLOG,415)
144  415 FORMAT(///6X,'TABLE OF SEASONAL CORRECTION FACTORS',
145     @ '/'0',4(' YRHR ',5X,' CF ',5X),
146     @ '/'+',4(' _____',5X,' _____',5X))
147  10  WRITE(KWLOG,420) (MNTH(I),CF(I),MNTH(I+6),CF(I+6),
148     @MNTH(I+12),CF(I+12),MNTH(I+18),CF(I+18),I=1,6)
149  11  CONTINUE
150  420 FORMAT(4(F8.0,5X,F8.4,5X))
151  C*****
152  C*****  CALCULATE THE SLOPES TO BE USED DURING INTERPLOATION
153  C*****
154     DO 40 I=1,24
155     IF(I-24 .NE. 0) THEN
156         CFDIF = CF(I+1) - CF(I)
157         MDIF= MNTH(I+1) - MNTH(I)
158         DSLOPE(I)=CFDIF/MDIF
159     ELSE
160         CFDIF=CF(1)-CF(I)

```

```

161         MDIF= YREND(MOYRS) - MNTH(I)
162         DSLOPE(I)=CFDIF/MDIF
163     END IF
164     40 CONTINUE
165 C***** -
166 C***** INITIALIZE THE PARAMETERS TO BE USED FOR READING THE DATA
167 C*****
168     IF (NPTS .LT. 200) THEN
169         NGO = NPTS
170         NLEFT = 0
171     ELSE
172         NGO = 200
173         NLEFT = NPTS
174     END IF
175 C*****
176 C***** READ IN BLOCK OF DATA; INCREMENT COUNTER
177 C*****
178     45 CONTINUE
179 c     READ(KRIES,425,END=900) (TT(IN),TTDTID(IN),TID(IN),PRS(IN)
180 c     @ TMP(IN),AMB(IN),TIME(IN),IN=1,NGO)
181 do 16 in=1,ngo
182 read(kries,425,end=900)tt(in),ttdtid(in),tid(in),time(in)
183 16 continue
184     425 FORMAT(4E15.7)
185     50 CONTINUE
186     NLEFT = NLEFT - NGO
187 C*****
188 C***** PROCESSING LOOP TO BE REPEATED FOR EACH SAMPLING PERIOD
189 C*****
190     DO 80 I=1,NGO
191         NTOTL = NTOTL + 1
192         YRHR=TIME(I)
193         DYRHR=DBLE(YRHR)
194 C*****
195 C***** IF IT IS THE FIRST TIME THROUGH LOOP,
196 C***** SEARCH TO FIND MONTH WHICH INCLUDES CURRENT YEARHOUR
197 C*****
198     IF (NTOTL .EQ. 1) THEN
199         DO 70 II=1,24
200         DEL = DYRHR-DBLE(MNTH(II))
201         IF (DEL .LE. 0) THEN
202             LOW = II-1
203             HI=II
204             DHMNTH=MNTH(HI)
205             DLMNTH=MNTH(LOW)
206             TDIF = LSTDEL
207             GO TO 75
208         ELSE IF (II .LT. 24) THEN
209             LSTDEL=DEL
210         ELSE
211             LOW = II
212             HI=25
213             DLMNTH=MNTH(LOW)
214             DHMNTH=YREND(MOYRS)

```

```

215         TDIF = DEL
216         GO TO 75
217     END IF
218     70 CONTINUE
219     75 CONTINUE
220 ELSE
221     DEL=DYRHR-DHMNTH
222     IF (DEL .GT. 0.0) THEN
223         LOW=LOW+1
224         IF(LOW .GT. 24) GO TO 999
225         DLMNTH=MNTH(LOW)
226         HI=HI+1
227         IF(HI .LE. 24) THEN
228             DHMNTH=DBLE(MNTH(HI))
229         ELSE
230             DHMNTH=YREND(NOYRS)
231         END IF
232     END IF
233     TDIF=DYRHR-DLMNTH
234 END IF
235 C*****
236 C***** INTERPOLATE TO CALCULATE SEASONAL CORRECTION FACTOR
237 C*****
238     SCF= DBLE(CF(LOW)) + TDIF*DBLE(DSLOPE(LOW))
239     TTSCF(I)= TT(I) + SCF
240     TDTSCF(I)=TTDTID(I) + SCF
241     80 CONTINUE
242 C*****
243 C***** WRITE BLOCK OF CORRECTED DATA TO DISK. IF END OF DATA,
244 C***** WRITE WRAP-UP INFO TO USER'S LOG.
245 C*****
246     WRITE(KWIES,425) (TTSCF(II),TDTSCF(II),TID(II),TIME(II),II=1,NGO)
247     IF(NFLAG .EQ. -1) GO TO 899
248     IF (NLEFT .LT. NGO) NGO = NLEFT
249     IF (NTOTL .LT. NPTS) GO TO 45
250     899 WRITE(KWLOG,430) NTOTL
251     430 FORMAT('0',I6,' CORRECTED TRAVEL TIMES WERE WRITTEN TO DISK')
252     WRITE(*,43)
253     STOP
254 C*****
255 C***** ERROR CONDITIONS:
256 C***** 1) END OF DATA ENCOUNTERED UNEXPECTEDLY
257 C*****
258     900 CONTINUE
259     NFLAG=-1
260     WRITE(KWLOG,436)
261     436 FORMAT(' FEWER POINTS THAN EXPECTED ON INPUT'/
262     @ ' NTOTL WILL BE REVISED')
263     NGO = IN - 1
264     GO TO 50
265 C*****
266 C***** 2) DATASET SPANS MORE THAN TWO CALENDAR YEARS
267 C*****
268     999 CONTINUE

```

```
269     WRITE(KWLOG,435)
270 435 FORMAT(' ***** RUN TERMINATED *****'/
271     ' DATASET EXCEEDED 24 MONTHS')
272     WRITE(*,43)
273 43  FORMAT(1X,/,5X,' YOUR TERMINAL CONTROL HAS NOW BEEN RETURNED.')
```

```
274     STOP
```

```
275     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.

```

1 C      THIS PROGRAM DOES TIDAL RESPONSE ANALYSIS (SEE MUNK & CARTWRIGHT,
2 C      SUBROUTINES AND FUNCTIONS FROM 'TIDELIB' ARE USED.
3 C      IMPLEMENTED ON THE PRIME BY DAVID LAI 10/20/80.
4 C      NO RESTRICTIONS ON THE length OF SERIES (PRIME HAS VIRTUAL
5 C      MEMORY). COMPUTATIONS ARE DONE ONLY ONCE THROUGH THE WHOLE SERIE
6 C      I.E. SERIES NOT CUT INTO SEGMENTS.
7 C      START TIME (SY), length OF SERIES (length), DELTA TIME (D)
8 C      NEED TO BE PUT IN.
9 C      DATA READ STATEMENTS REQUIRED CHANGES ACCORDING TO DATASET.
10 C     OPTIONS ARE SET BY DATA STATEMENTS:
11 C     *** FOLLOWING VALUES .EQ. 1 INDICATE::
12 C     IADJD -- SUBTRACT MEAN FROM SERIES AND MULTIPLY BY CALIB.
13 C     IDADM -- COMPUTE ADMITTANCE
14 C     IDRSP -- CREATE PREDICTED TIDE SERIES
15 C     IRESID - COMPUTE RESIDUAL SERIES
16 C     *** FOLLOWING VALUE .EQ.0 INDICATES ::
17 C     KOMPLX -- ONLY REAL PART OF PREDICTED SERIES IS TO BE RETAINED.
18 C     *** FOLLOWING VALUES .LE.0 INDICATE::
19 C     IPRIND -- PRINT PARTIAL LIST OF NORMALIZED INPUT DATA
20 C     IPRINP -- PRINT PARTIAL LIST OF PREDICTED TIDE
21 C     IPRINR -- PRINT PARTIAL LIST OF RESIDUAL SERIES
22 C     **** OTHERWISE THE WHOLE SERIES ARE PRINTED ***
23 C     *** FOLLOWING QUANTITIES DEPEND ON TYPE OF TIDES INVOLVED.
24 C     *** SEE LISTINGS OF TIDELIB SUBROUTINES FOR DETAILS ****
25 C     LGAMMA, NUMGMN, MORDER, NDEGRE, JP, KP, HH, KH, NP, WPHGP1, NH.
26 C*****
27 C
28 C     XXM1 DIMENSIONED AT LEAST 4*NPH*(NPH+1) WHERE NPH IS THE NUMBER
29 C     OF P,H COMBINATIONS.
30 C
31 C     C DIMENSIONED ATLEAST (2*length)+1000
32 C
33 C     Y DIMENSIONED (length)  WHERE length IS length OF TIME SERIES
34 C
35 C     YPRED DIMENSIONED (length)
36 C
37 C     I/O FILES/PARAMETERS
38 C     =====
39 C***** KCTRL (UNIT 13) - CONTROL FILE
40 C***** KOUT  (UNIT 15) - RESPO OUTPUT
41 C***** KIN   (UNIT 16) - DESPIKED INPUT
42 C
43 C=====
44     parameter ( n_dim1 = 25000 )
45     parameter ( n_dim2 = 2 * n_dim1 + 1000 )
46     PARAMETER(KCTRL=13,KOUT=15,KIN=16)
47     COMPLEX C(n_dim2)
48     COMMON /DUM1/ C
49     COMMON /DUM2/ Y(n_dim1)

```



```

50     COMMON /DUM3/ YPRED(n_dim1)
51     COMMON /DUM4/ XX1(728) ! nph = 13
52     COMMON /DUM5/ YRHR(n_dim1)
53     COMMON /DUM6/ YRES(n_dim1)
54 C*****
55     COMMON /RESTOR/ IWISHR,SXR,EXR,SYR,EYR,DR,LIMPR,KOMPLX
56     COMMON /TEAPOT/ LEQPOT,LF,THETA0,PHIO,SD,DD,ED,IWISHT
57     COMMON /WAITER/ IWISHL,IFINAL,SX,EX,SY,EY,D,LIMP
58     DOUBLE PRECISION YSUM
59     DOUBLE PRECISION SD,DD,ED,SY,Y,EY,yearh-
60     DOUBLE PRECISION XXM
61     DIMENSION LGAMMA(10),MORDER(10),NDEGRE(10),PHWTS(80),HH(10),
62 + JP(10),H(10),NP(10),NH(10)
63     DIMENSION GRAY(20),NCONST(10)
64     DIMENSION ITITLD(4),ITITLP(4),ITITLR(4)
65     DIMENSION XXM(1)
66     CHARACTER*40 HEADR,FORM
67     REAL *8 ANAME(5),SAM
68     integer leap,spread
69     EQUIVALENCE(XXM1(1),XXM(1))
70     NAMELIST/CARD1/HEADR
71     NAMELIST/CARD2/FORM
72     NAMELIST/CARD3/length,year,yearhr,D
73 C*****
74     DATA ANAME/'WYY-1 ','POTTY ','WEIHTY ','TADM ','SPONTY '/
75     DATA ITITLD/4H1NOR,4HMALI,4H2ED ,4HDATA/
76     DATA ITITLP/4H1PRE,4HDICT,4HED T,4HIDE /
77     DATA ITITLR/4HOTID,4HE RE,4HSIDU,4HAL /
78     DATA IDWTS/ 1/
79 C   IADJD.WE.0 MEANS REMOVE MEAN FROM SERIES AND MULTIPLY BY CALIB.
80     DATA IADJD/1/, CALIB/1.0/, IPRIND/-1/
81     DATA IDADM/1/
82     DATA IDRSP/1/ , KOMPLX/0/ , IPRINP/-1/
83     DATA IRESID/1/ , IPRINR/-1/ , RSCALE/1E3/
84     DATA LGAMMA /3,3, 8*0/ , NUNGMN/2/
85     DATA MORDER /1,2, 8*0/
86     DATA NDEGRE /2,2, 8*c/
87     DATA JP /1,2, 8*0/ , KP/2/
88     DATA HH / -48. , 0. , 48. , 7*0.0 / , KH/3/
89 C   DATA HH /-96. ,-48. ,0. ,48. ,96. ,5*0.0/ , KH/5/
90     DATA NP/0,2, 8*0/ , NPHGP1/2/
91     DATA NH/0,3, 8*0/
92 C   DATA NH/0,5,8*0/
93 C*****
94     HCORR(HHH)=FLOAT(ROUND(HHH/D))*D
95 ! 900  FORMAT('0',F9.6,11F10.6)
96     900  FORMAT('0',6f15.6)
97     90  FORMAT(12F6.2)
98     91  FORMAT(4A4//((6(F10.3,F8.?)))
99     81  FORMAT('OSYY= ',D20.10,' length=',I10 ' D=',D20.10,' EYY=',D20.10)
100    910  FORMAT(9H1ISKIPR =,I6)
101     2  FORMAT(3H0 ,A6,F6.0)
102    443  FORMAT(8HOPHPTS =)
103    444  FORMAT(F6.0,F8.2,2F11.6)

```

```

104 C*****
105 C   *** INSERT SYY=START TIME IN HOURS FROM BEGINNING OF THIS CENTURY
106 C   ***           length=length OF SERIES
107 C   ***           D=DELTA TIME IN HOURS
108   open(UNIT=kctrl,status='old',FORM='FORMATTED',READONLY)
109   read(kctrl,NML=CARD1)
110   read(kctrl,NML=CARD2)
111   READ(KCTRL,NML=CARD3)
112 C*****
113 C***** Conversion from year & year hour to Year Hour from 1900
114 C*****
115 Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
116 C***** correction to conversion. Spread was previously defined as
117 C*****   spread=year-1900
118 C***** The subtraction of unity insures the proper treatment of a leap
119 C***** year. This was verified with Dr Wimbush's Kalday function.
120 C*****
121 C***** A negative yearhr error trap was removed. Karen and I were
122 C***** able to justify why positive year hours were necessary.
123 C*****
124 C*****   E.Fields 9-Feb-90
125 Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
126   spread=year-1900-1
127   leap=int(spread/4.)
128   syy=(spread+1)*8760+float(leap)*24+yearhr
129   print *,HEADR,FORM
130   print *,length,year,yearhr
131 C   if (yearhr.LT.0.0) then
132 C     print 41,headr
133 C 41   format(1x,a40,/,5x,
134 C   @     'error in input yearhr. It must be positive!')
135 C     call exit
136 C   endif
137 C   ***** END OF INPUT
138 C*****
139   EYY=SYY+DBLE(FLOAT(length-1))*D
140   PRINT 81,SYY,length,D,EYY
141   H(1)=HCORR(HH(1))
142   HMIN=H(1)
143   HMAX=H(1)
144   DO 820 I=2,KH
145     H(I)=HCORR(HH(I))
146     IF(H(I).LT.HMIN) HMIN=H(I)
147     IF(H(I).GT.HMAX) HMAX=H(I)
148 820 CONTINUE
149 C
150   *** TIDPOT ***
151   LEQPOT=1
152   LF=0
153   THETAO=0.
154   PHIO=0.
155   SD=SYY+DBLE(HMIN)
156   DD=D
157   ED=EYY+DBLE(HMAX)

```

```

158     IWISHT=1
159     TRY=POTTY(LGAMMA,MORDER,NDEGRE,NUMGMW, C,NTIM)
160     SAM=AWAME(2)
161     PRINT 2,SAM,TRY
162     IF(TRY.EQ.0.) GO TO 40
163     CALL EXIT
164     40 CONTINUE
165     PRINT 769, NTIM
166     769 FORMAT('ONTIM =',I6)
167     PRINT 336, (C(I), I=1,12)
168     !     PRINT 336, (C(I), I=1,200)
169     336 FORMAT(27H0-TIDPOT SERIES BEGINNING -,(/4(5X,2F12.6)))
170     C
171     C     *** TIDWTS ***
172     C
173     C     *** CHECK TO READ IN WEIGHTS ***
174     IF(IDWTS.GT.0) GO TO 48
175     KPS=1
176     KPHWTS=4*MAX0(KPS,-IDWTS)
177     READ 444, (PHWTS(I),I=1,KPHWTS)
178     48 IWISHL=1
179     IFINAL=1
180     SY=SY
181     EY=EY
182     D=DD
183     SX=SD
184     EX=ED
185     LIMP=NUMGMW
186     IF(IDWTS.LE.0 .AND. (IDRSP.LE.0 .OR. IRESID.LE.0)) GO TO 106
187     C*****
188     C
189     C
190     C     ***** READ IN DATA SERIES *****
191     C
192     OPEN(UNIT=kin,status='old',FORM='FORMATTED',READONLY)
193     READ(KIN,FORM) (Y(I), yrhr(i), I=1,length)
194     !     DO nn = 1 , length
195     !         y(nn) = y(nn) * 100.
196     !     ENDDO
197     C
198     C
199     C     ***** END OF READ DATA *****
200     C
201     C*****
202     NY=INT(1.5+(EY-SY)/D)
203     PRINT 770, NY
204     770 FORMAT('ONY =',I6)
205     PRINT 900, (Y(I), I=1,12)
206     NYM11=NY-11
207     PRINT 900, (Y(I), I=NYM11,NY)
208     YSUM=0.
209     DO 70 I=1,NY
210         YSUM=YSUM+DBLE(Y(I))
211     70 continue

```

```

212     YAVE=YSUM/DBLE(FLOAT(NY))
213     PRINT 771, YAVE
214 771  FORMAT('OYAVE =',E13.7)
215     KUP=MINO(NY,500)
216     IF(IADJD.LE.0) GO TO 95
217 C     REMOVE MEAN AND NORMALIZE
218     PRINT 775, CALIB
219 775  FORMAT('OCALIB =',E12.4)
220     DO 80 I=1,NY
221     Y(I)=(Y(I)-YAVE)*DBLE(CALIB)
222 80   continue
223     NNY=NY
224     IF(IPRIND .LE. 0) NNY=KUP
225     PRINT 91, ITITLD,(Y(I), I=1,NNY)
226 95   IF(IDWTS.LE.0) GO TO 106
227     PRINT 8006, (NP(I), I=1,NPHGP1)
228 8006  FORMAT('O NP',30I4)
229     TRY=WEIHTY(C, Y, JP,KP, H,KH, NP,NH,NPHGP1, XXM1,XXM,
230 + PHWTS,KPHWTS)
231     SAM=AWANE(3)
232     PRINT 2,SAM,TRY
233     IF(TRY.EQ.0.) GO TO 100
234     CALL EXIT
235 100  IWISHL=0
236 106  PRINT 443
237     PRINT 444, (PHWTS(I), I=1,KPHWTS)
238 C
239 C   *** TIDADM ***
240     IF(IDADM .LE. 0) GO TO 220
241     KEEP=1
242     DO 200 IC=1,NUMGMH
243     CMPNT=FLOAT(IC)
244     IF(LGAMMA(IC).EQ.4) GO TO 190
245     IF(NDEGRE(IC).GT.2) GO TO 3000
246     IF(MORDER(IC).EQ.0) GO TO 2500
247     DELF=0.0366011
248     IF(MORDER(IC).EQ.2) GO TO 2000
249     FS=0.8929346
250     NCONST(1)=2
251     NCONST(2)=4
252     NNC=2
253 165  NTYPE=10
254     LO=10
255     TRY=TADM(PHWTS,KPHWTS, ORAY,LO, FS,DELF, CMPNT,KEEP)
256     CALL HG(ORAY,LO, NCONST,NNC, MORDER(IC), NTYPE)
257     SAM=AWANE(4)
258     PRINT 2,SAM,TRY
259     IF(TRY.EQ.0.) GO TO 170
260     CALL EXIT
261 170  DELF=0.104018
262     FS=0.8932441
263     IF(MORDER(IC).EQ.2) FS=1.895982
264     NTYPE=10
265 175  LO=4

```

```

266 180 TRY=TADM(PHWTS,KPHWTS, ORAY,LO, FS,DELF, CMPNT,KEEP)
267      NCONST(1)=1
268      NCONST(2)=2
269      NNC=2
270      CALL HG(ORAY,LO, NCONST,NNC, MORDER(IC), NTYPE)
271      SAM=ANAME(4)
272      PRINT 2,SAM,TRY
273      IF(TRY.EQ.0.) GO TO 200
274      CALL EXIT
275 190  DELF=0.005475819
276      FS=0.9972620907
277      IF(NDEGRE(IC).EQ.1) FS=1.0
278      IF(MORDER(IC).EQ.2) FS=2.0
279      NTYPE=-IFIX(2.*ABS(1.-FS)+0.999)
280      LO=4
281      GO TO 180
282 2000 FS=1.8590714
283      NCONST(1)=3
284      NCONST(2)=5
285      NNC=2
286      GO TO 165
287 2500 FS=0.00547582
288      DELF=0.06772639
289      NTYPE=-10
290      GO TO 175
291 3000 IF(MORDER(IC).NE.3) GO TO 3300
292      DELF=0.036291647
293      FS=2.862118775
294 3200 NTYPE=IROUND(FS)
295      LO=4
296      GO TO 180
297 3300 DELF=0.073202204
298      FS=0.9664462631
299      IF(MORDER(IC).EQ.2) FS=1.895672514
300      GO TO 3200
301 200 CONTINUE
302      PRINT 8006, (NP(I), I=1,NPHGP1)
303 C
304 C *** TIDRSP ***
305 220 IF(IDRSP.LE.0) GO TO 160
306      LIMPR=LIMP
307      NP(2)=KP
308 C ***** NP(2)=1 -- PREDICTED TIDES CONSIST OF ONLY SEMI-DIURNAL **
309 C NP(2)=1
310 C *****
311      NPGP1=2
312      IWISHR=1
313      SIXR=SI
314      EXR=EX
315      SYR=SY
316      EYR=EY
317      DR=D
318      TRY=SPONTY(C, PHWTS,KPHWTS, JP, NP,NPGP1, YPRED,KY)
319 C ***** FOLLOWING STATEMENT COMPUTES ONLY SEMI-DIURNAL TIDES ***

```

```

320 C   TRY=SPONTY(C, PHWTS,KPHWTS, 2, NP,2, YPRED,KY)
321     SAM=ANAME(5)
322     PRINT 2,SAM,TRY
323     IF(TRY.EQ.0.) GO TO 130
324     CALL EXIT
325 130  CONTINUE
326     PRINT 772, KY
327 772  FORMAT('OKY  =',I6)
328     KOMP=MINO(2,IABS(KOMPLX)+1)
329     KYY=KOMP*KY
330     KUP=MINO(KY,500)
331     NNY=KY
332     IF(IPRINP .LE. 0) NNY=KUP
333     NNY=KOMP>NNY
334     PRINT 91, ITITLP,(YPRED(I), I=1,NNY)
335 150  IF(IRESID .LE. 0) GO TO 160
336     IF(KOMPLX .NE. 0) GO TO 153
337     DO 152 I=1,KY
338         YRES(I)=Y(I)-YPRED(I)
339 152  continue
340     GO TO 155
341 153  DO 154 I=1,KY
342         YRES(I)=Y(I)-YPRED(2*I-1)
343 154  continue
344 156  NNY=KY
345     IF(IPRINR .LE. 0) NNY=KUP
346     ISKIPR=MAXO(1,IPRINR)
347     PRINT 910, ISKIPR
348     PRINT 91,ITITLR,(YRES(I), I=1,NNY,ISKIPR)
349 160  CONTINUE
350 C*****
351 C
352 C   *** WRITE RESIDUAL TIME SERIES ****
353 C
354     OPEN(UNIT=KOUT,status='new',FORM='FORMATTED')
355     do 13 i=1, ky
356     WRITE(KOUT,162) y(i),yres(i),,pred(i),yrhr(i)
357 13  continue
358 162  FORMAT(4E15.7)
359     CLOSE(KIN)
360     close(kctrl)
361     CLOSE(KOUT)
362 C
363 C*****
364     CALL EXIT
365     END
1     FUNCTION PTTY(LGAMMA,MORDER,NDEGRE,NUMGMW, C,NTIM)
2
3 C
4 C TITLE - PTTY = POTENTIAL TYDE
5 C   GENERATION OF TIDE POTENTIALS
6 C
7 C
8 C   ---ABSTRACT---

```

```

9 C
10 C IF LEQPOT .NE. 0 PTTY(LGAMMA,MORDER,NDEGRE,NUMGMN,
11 C C,NTIM) GENERATES FUNCTIONS RELATED TO THE TIDE
12 C POTENTIALS. THE FUNCTIONS ARE NOT COMPUTED AS A SUPER-
13 C POSITION OF TIME HARMONICS IN THE CLASSICAL SENSE, BUT
14 C DIRECTLY FROM THE KNOWN ORBITAL CONSTANTS.
15 C (SEE W.E. MUNK AND D.E. CARTWRIGHT, 1966. TIDAL SPECTRO-
16 C SCOPE AND PREDICTION, PHIL. TRANS. ROY. SOC. A, 259,
17 C 533-581)
18 C
19 C IF LEQPOT=0 PTTY(LGAMMA,MORDER,NDEGRE,NUMGMN,C)
20 C GENERATES THE GAMMA EQUILIBRIUM TIDE AT A
21 C PLACE THETA,PHI (IN DEGREES) AS DERIVED FROM THE FUNDA-
22 C MENTAL DEFINITIONS WITHOUT EXPANSION INTO SPHERICAL
23 C HARMONICS.
24 C
25 C
26 C --STATISTICS--
27 C
28 C LANGUAGE - FORTRAN IV (CDC3600,B6500)
29 C EQUIPMENT - NO SPECIAL REQUIREMENTS
30 C STORAGE - 610(OCTAL) = 392(DECIMAL) LOCATIONS
31 C SPEED -
32 C AUTHOR - MARK WIMBUSH IGPP JUL 1970
33 C LAST MOD - MARK WIMBUSH NOVA APR 1972
34 C CATAGORIES
35 C STATUS -
36 C
37 C LIBRARY ROUTINES USED - AMENPI, RECURQ, SHMIDT, SETUPM, ORBITS
38 C SYSTEM ROUTINES USED - Q2Q07110,Q1Q00310,Q1Q04310,Q1Q04330,Q1Q02330,
39 C DMOD,XINTF,SQRTF,SINF,COSF,AIMAG
40 C
41 C
42 C ----USAGE----
43 C
44 C SAMPLE CALL
45 C J = PTTY(LGAMMA,MORDER,NDEGRE,NUMGMN, C,NTIM)
46 C
47 C NOTE - DIMENSION OF ARRAY C IS NUMGMN*NTIM
48 C (COMPLEX UNLESS LEQPOT=0)
49 C
50 C
51 C INPUTS
52 C
53 C LGAMMA(I) =1 FOR MOON'S GRAVITATIONAL POTENTIAL
54 C =2 FOR SUN'S GRAVITATIONAL POTENTIAL
55 C =3 FOR TOTAL GRAVITATIONAL POTENTIAL
56 C =4 FOR SUN'S RADIATIONAL POTENTIAL
57 C
58 C MORDER(I) VALUE OF M IN ITH TRIPLET
59 C (ORDER OF SPHERICAL HARMONIC)
60 C NOT MEANINGFUL IF LEQPOT=0
61 C
62 C NDEGRE(I) VALUE OF N IN ITH TRIPLET

```

63 C (DEGREE OF SPHERICAL HARMONIC)
 64 C NOT MEANINGFUL IF LEQPOT=0
 65 C
 66 C NUMGMN NUMBER OF GAMMA,M,N TRIPLETS
 67 C
 68 C ----COMMON /TEAPOT/
 69 C
 70 C LEQPOT =0 IF THE EQUILIBRIUM TIDE IS TO BE STORED AT C.
 71 C LF AND ARRAYS MORDER AND NDEGRE ARE NOT MEANINGFUL
 72 C AND ARE NOT CHECKED.
 73 C .WE. 0 IF TIDE POTENTIALS ARE TO BE STORED AT C.
 74 C
 75 C LF =0 IF THETAO AND PHIO ARE TO BE IGNORED
 76 C NOT MEANINGFUL IF LEQPOT=0
 77 C
 78 C THETAO LOCAL COLATITUDE, IN DEGREES
 79 C (NEEDED ONLY IF LF .WE. 0 OR IF LEQPOT=0)
 80 C
 81 C PHIO GREENWICH EAST LONGITUDE, IN DEGREES
 82 C (NEEDED ONLY IF LF .WE. 0 OR IF LEQPOT=0)
 83 C
 84 C SD START TIME IN HOURS (DOUBLE PRECISION)
 85 C
 86 C DD INCREMENT TIME IN HOURS (DOUBLE PRECISION)
 87 C
 88 C ED END TIME IN HOURS (DOUBLE PRECISION)
 89 C
 90 C NOTE. ZERO TIME IS TAKEN TO BE 1900 JAN 1 0000HRS GMT
 91 C
 92 C INISHL MUST BE SET NON-ZERO IN INITIAL CALL OF POTTY.
 93 C IN FURTHER CALLS, INISHL MAY BE SET TO ZERO IF THE ONLY
 94 C OTHER CHANGED INPUT PARAMETERS ARE SD, DD, AND ED
 95 C
 96 C
 97 C OUTPUTS
 98 C
 99 C POTTY =0. IF INPUT ITEMS HAVE VALID VALUES
 100 C =1. IF MORDER .GT. NDEGRE
 101 C =2. IF NDEGRE .LT. 1
 102 C =3. IF MORDER .LT. 0
 103 C =4. IF LGAMMA .LT. 1 OR LGAMMA .GT. 4
 104 C =5. IF NUMGMN .LT. 1
 105 C =6. IF IMPOSSIBLE COLATITUDE (THEATO)
 106 C =7. IF INVALID TIME GROUP (SD,DD,ED)
 107 C
 108 C C IF LEQPOT .WE. 0, THE MERGED COMPLEX SERIES C OR F OF
 109 C TIDE POTENTIALS ACCORDING AS LF=0 OR 1 (SEE EQU. A7
 110 C OF MUNK AND CARTWRIGHT OR DESCRIPTION OF BOOM STATEMENT
 111 C TIDPOT)
 112 C IF LEQPOT = 0, THE GAMMA EQUILIBRIUM TIDE F(THETAO,PHIO)
 113 C AS DERIVED FROM THE FUNDAMENTAL DEFINITIONS WITHOUT
 114 C EXPANSION INTO SPHERICAL HARMONICS.
 115 C
 116 C NTIM NUMBER OF DIFFERENT TIMES FOR WHICH TERMS OF C ARE


```

117 C          COMPUTED - I.E. INT(1.5+(ED-SD)/DD)
118 C
119 C
120 C EXAMPLES _
121 C
122 C
123 C PROGRAM FOLLOWS BELOW
124 C
125
126 !          implicit real*8 (a-h,o-z)
127
128          COMMON /TEAPOT/ LEQPOT,LF,THETA0,PHI0,SD,DD,ED,INISHL
129          COMMON /WORKIN/ XNPI(138),RADN(16),DEIMLS(17),DEIMLM(17),
130          1 DEIMLF(17),REQ(285),QS(153),QM(153),QF(153)
131          COMMON /POTLUK/ LAST,F,CTHETA,STHETA,CZ(2),SZ(2),BIGL(2),RBDR(2),
132          1 TD,HR
133 c//////////
134          real*8 SD,DD,ED,TD,TGP12D
135 !          real*8 pax,amdme,pid180
136          COMPLEX DEIMLS,DEIMLM,DEIMLF,DK,CK
137          DIMENSION LGAMMA(NUMGMN),MORDER(NUMGMN),NDEGRE(NUMGMN),
138          1 C(1),PAX(2),AMDME(2)
139 C          (SEE @NOTE@ ABOVE FOR TRUE DIMENSION OF C)
140          EQUIVALENCE (IGMN,XNPI(1))
141 C          1964 I.A.U. ASTRONOMICAL CONSTANTS (SEE AMERICAN EPHEMERIS AND
142 C          NAUTICAL ALMANAC 1971, P.481)
143 C          SOLAR PARALLAX = 8@.794
144 C          (8.794/3600.)*(PI/180.)=4.2634515117E-5
145 C          SINE PARALLAX FOR MOON = 3422@.451
146 C          (3422.451/3600.)*(PI/180.)=1.6592510677E-2
147 C          AEQUATORIAL = 6378160 M
148 C          MSUN/(MEARTH+MMOON) = 328912 , MEARTH/MMOON = 81.30
149 C          (6378160.*100.CH)*328912.*(1.+1./81.30)=2.1236572163E14 CM
150 C          (6378160.*100.CH)/81.30=7.8452152522E6 CH
151          DATA PAX /4.2634515117e-5, 1.6592510677e-2/,
152          1 AMDME /2.1236572163e14, 7.8452152522e6/,
153          2 PID180 /.01745329252e0/
154
155 C . PTTY IS THE MAIN INDEPENDENT SUBPROGRAMME FOR STATEMENTS
156 C @TIDPOT@ AND @TIDEQU@. (PROGRAMME AND ASSOCIATED SUBROUTINES
157 C WRITTEN BY MARK WIMBUSH - JULY 1970)
158
159          PTTY=0.
160          IF(SD.NE.ED) GO TO 10
161          NTIM=1
162          GO TO 30
163          10 DDEMSD=SNGL(DD/(ED-SD))
164          IF(DDEMSD.GT.0.) GO TO 20
165          PTTY=7.
166          GO TO 40
167          20 NTIM=INT(1./DDEMSD+1.5)
168 C          SKIP NEXT SECTION IF NOT INITIAL CALL
169          30 IF(INISHL.EQ.0) GO TO 70
170          PHI=PHI0+PID180

```

```

171      THETA=THETAO+PID180
172      CTHETA=COS(THETA)
173      STHETA=SIN(THETA)
174      IF(NUMGMN.LT.1) PTTY=5.
175      IF(STHETA.LT.0..AND.(LEQPOT.EQ.0.OR.LF.WE.0)) PTTY=6.
176  40  IF(PTTY.WE.0.) RETURN
177  C   COMPUTE NMAX (MAXIMUM N), NRMAX (MAXIMUM RADIATIONAL N),
178  C   FORM LAST ACCORDING TO DESCRIPTION ABOVE, AND CHECK FOR INVALID
179  C   GAMMA,M,N TRIPLETS (ASSIGNING VALUES TO PTTY ACCORDING TO
180  C   DESCRIPTION ABOVE)
181      NMAX=0
182      NRMAX=0
183      LAST=0
184      MAST=-1
185
186      DO 60 IGMN=1,NUMGMN
187      LG=LGAMMA(IGMN)
188      M=MORDER(IGMN)
189      N=NDEGRE(IGMN)
190      IF(LEQPOT.EQ.0) GO TO 50
191      IF(N.GT.NMAX) NMAX=N
192      IF(LG.EQ.4.AND.N.GT.NRMAX) NRMAX=N
193      IF(M.GT.N) PTTY=1.
194      IF(N.LT.1) PTTY=2.
195      IF(M.LT.0) PTTY=3.
196  50  IF(LG.LT.1.OR.LG.GT.4) PTTY=4.
197  C   IF BAD GAMMA,M,N TRIPLET IS FOUND, RETURN
198      IF(PTTY.WE.0.) RETURN
199      IF(LG.WE.1) MAST=1
200      IF(LG.WE.2.AND.LG.WE.4) LAST=1
201  60  CONTINUE
202
203      LAST=LAST*MAST
204      NMAXP1=NMAX+1
205
206  C--- SKIP NEXT SECTION IF STATEMENT IS @TIDEQU@
207
208      IF(LEQPOT.EQ.0) GO TO 70
209
210  C--- CALCULATE TIME INDEPENDENT FACTORS, ALSO CALCULATE COEFFICIENTS
211  C--- FOR RECURSION FORMULA USED TO GENERATE SCHMIDT FUNCTION PART
212  C--- OF SPHERICAL HARMONICS
213
214      CALL AMENPI(PAX,AMDME, LGAMMA,NDEGRE,NUMGMN, NRMAX)
215      CALL RECURQ(NMAX)
216      F=0.
217
218  C--- SKIP NEXT SECTION IF @F@ IS NOT SET
219
220      IF(LF.EQ.0) GO TO 70
221
222  C--- COMPUTE COMPONENTS OF SPHERICAL HARMONICS AT THE LOCATION
223  C--- THETA,PHI
224

```

```

225     CALL SHMIDT(NMAX,CTHETA,STHETA, QF)
226     F=1.
227     CALL SETUPM(PHI, DEIMLF,NMAXP1)
228     F=-F
229     70 K=1 -
230
231 C--- SD(DD)ED ARE START(INTERVAL)END TIMES IN HOURS SINCE
232 C--- 1900 JAN 1 0000 HRS GMT
233
234     TGP12D=SD+12D0
235     DO 260 ITIM=1,NTIM
236         HR=SNGL(DMOD(TGP12D,24D0))
237 C--- TD IS TIME IN JULIAN CENTURIES SINCE 1899 DEC 31 NOON GMT
238     TD=TGP12D/87660D0
239 C--- CALCULATE NEEDED ORBITAL PARAMETERS OF SUN AND MOON
240     CALL ORBITS
241 C--- BRANCH IF STATEMENT IS @TIDEQU@
242     IF(LEQPOT.EQ.0) GO TO 200
243 c//////////
244 C--- SKIP NEXT SECTION IF ALL GAMMAS ARE @MG@
245     IF(LAST.EQ.-1) GO TO 80
246 C--- COMPUTE SPHERICAL HARMONIC COMPONENTS OF GREENWICH COORDINATES
247 C--- OF SUN
248     CALL SHMIDT(NMAX,CZ(1),SZ(1), QS)
249     CALL SETUPM(BIGL(1), DEIMLS,NMAXP1)
250 C--- SKIP NEXT SECTION IF ALL GAMMAS ARE @SG@ OR @RA@
251     IF(LAST.EQ.0) GO TO 90
252
253 C--- COMPUTE SPHERICAL HARMONIC COMPONENTS OF GREENWICH COORDINATES
254 C--- OF MOON
255
256     80 CALL SHMIDT(NMAX,CZ(2),SZ(2), QM)
257
258     CALL SETUPM(BIGL(2), DEIMLM,NMAXP1)
259 c//////////
260     90 II=1
261     DO 190 I=1,NUMGM
262         MP1=MORDER(I)+1
263         NP1=NDEGRE(I)+1
264         J=NP1*(NP1-1)/2+MP1
265         LG=LGAMMA(I)
266         CK=(0.,0.)
267     100 KK=XNPI(II)
268         IF(XK.EQ.0.) GO TO 180
269         GO TO (160,130,120,110,150), LG
270     110 RK=RBDR(1)**2
271         GO TO 140
272     120 LG=5
273     130 RK=RBDR(1)**NP1
274     140 QK=QS(J)
275         DK=DEIMLS(MP1)
276         GO TO 170
277     150 LG=0
278     160 RK=RBDR(2)**NP1

```

```

279         QK=QM(J)
280         DK=DEIMLM(MP1)
281 C---      COMPUTE C = A + IB
282 170      CK=CK+IK*RK+QK*DK
283 180      II=II+1 -
284         IF(LG.EQ.5) GO TO 100
285 C---      IF @F@ IS SET MULTIPLY BY VALUE OF SPHERICAL HARMONIC AT
286 C---      LOCATION THETA,PHI
287         IF(LF.NE.0) CK=QF(J)*DEIMLF(MP1)*CK
288         C(2*K-1)=REAL(CK)
289         C(2*K)=AIMAG(CK)
290         IF(MP1.EQ.1) C(2*K)=1.
291 190      K=K+1
292         GO TO 160
293 c//////////
294 C      STATEMENT IS @TIDEQU@ - COMPUTE EQUILIBRIUM TIDE AT THETA,PHI
295 200      DO 250 I=1,NUNGM
296         SFK=0.
297         LSM=1
298         IF(LGAMMA(I).EQ.1) LSM=2
299 210      CALFA=CTHETA+CZ(LSM)+STHETA*SZ(LSM)*COS(PHI-BIGL(LSM))
300         IF(LGAMMA(I).EQ.4) GO TO 220
301         P=PAX(LSM)*RBDR(LSM)
302         IF(LSM.NE.2) GO TO 230
303         SFK=SFK+AMDME(2)*P*(1./SQRT((P-2.*CALFA)*P+1.))-1.-P*CALFA)
304         GO TO 240
305 220      RK=RBDR(1)**2
306         SFK=-.25*RK
307         IF(CALFA.GT.0.) SFK=SFK+CALFA*RK
308         GO TO 240
309 230      GAM=(2.*CALFA-P)*P
310         SFK=AMDME(1)*P*(((.2734375+GAM+.3125)*GAM+.375)*
311 +         GAM**2-.5*P**2)
312         IF(LGAMMA(I).NE.3) GO TO 240
313         LSM=2
314         GO TO 210
315 240      C(K)=SFK
316 250      K=K+1
317
318 C---      IF END TIME NOT REACHED, GO BACK AND DO CALCULATIONS FOR
319 C---      NEXT TIME STEP
320
321 260      TGP12D = TGP12D + DD
322
323         RETURN
324
325         END
326 1         FUNCTION WEIHTY(X, Y, JP,KP, H,KH, NP,NH,NPHGP1, XXM1,XXM,
327 2         1 PHWTS,KPHWTS)
328 3 C
329 4 C
330 5 C TITLE - WEIHTY = WEIGHTS, TYDE
331 6 C     GENERATES OPTIMUM PREDICTION WEIGHTS
332 7 C

```

```

8 C
9 C          ---ABSTRACT---
10 C
11 C          WEIHTY COMPUTES COMPLEX WEIGHTS W(P,H) SUCH THAT THE
12 C          - REAL PART OF THE SUM OVER ALL SPECIFIED P,H COMBINA-
13 C          TIONS OF CONJG(W(P,H))*X(P,T+H) IS AS CLOSE AS POS-
14 C          SIBLE TO Y(T) IN THE LEAST-SQUARES SENSE. X(P,T) RE-
15 C          PRESENTS THE PTH COMPONENT OF THE COMPLEX REFERENCE
16 C          SERIES X AT TIME T. Y(T) REPRESENTS THE VALUE AT TIME
17 C          T OF THE SERIES TO BE PREDICTED. THE P AND H VALUES
18 C          TO BE USED ARE GIVEN BY ARRAYS JP AND H, AND THE P,H
19 C          COMBINATIONS ARE SPECIFIED BY ARRAYS NP AND NH - FOR
20 C          EACH N SUCH THAT 0 .LT. N .LE. NPHGP1, ALL JP(J)
21 C          SUCH THAT NP(N-1) .LT. J .LE. NP(N) ARE COMBINED WITH
22 C          ALL H(I) SUCH THAT NH(N-1) .LT. I .LE. NH(N) . THE OUT-
23 C          PUT ARRAY PHWTS CONSISTS OF P, H, W(P,H) MERGED
24 C          (I.E. A 4 REAL COMPONENT SERIES).
25 C
26 C
27 C          --STATISTICS--
28 C
29 C LANGUAGE - UCSD FORTRAN 63
30 C EQUIPMENT - NO SPECIAL REQUIREMENTS
31 C STORAGE - 484 WORDS FOR THIS PROGRAMME + 436 WORDS FOR
32 C          ASSOCIATED SUBPROGRAMMES (MAVDUB,ISMI,INTCH) +
33 C          1018 WORDS COMMON + SPACE FOR ARRAYS X, Y, JP, H,
34 C          NP, NH, XXM1, PHWTS (I.E. 2*KX+KY+KP+KH+2*NPHGP1+
35 C          KM1+KPHWTS WORDS)
36 C SPEED -
37 C AUTHOR - MARK WIMBUSH IGPP JUL 1970
38 C LAST MOD - MARK WIMBUSH NOVA APR 1972
39 C CATAGORIES -
40 C STATUS -
41 C
42 C LIBRARY ROUTINES USED- MAVDUB, ISMI
43 C SYSTEM ROUTINES USED - IROUND, XINTF, FLOATF, Q8QINGOT, Q8QENGOT,
44 C          Q8QGOTTY
45 C
46 C
47 C          ----USAGE----
48 C
49 C SAMPLE CALL
50 C          J = WEIHTY(X,Y,JP,KP,H,KH,NP,NH,NPHGP1,XXM1,XXM, PHWTS,KPHWTS)
51 C
52 C NOTE - XXM1(1),XXM(1) SHOULD BE EQUIVALENCED PRIOR TO
53 C          CALLING WEIHTY
54 C          - DIMENSION OF REFERENCE SERIES X IS
55 C          LIMP*INT(1.5+(EX-SX)/D) (COMPLEX)
56 C          - DIMENSION OF DATA SERIES Y IS KY=INT(1.5+(EY-SY)/D)
57 C          - DIMENSION OF WORKING STORAGE ARRAY XXM1 IS
58 C          4*NPH*(NPH+1), WHERE NPH IS THE TOTAL NUMBER
59 C          OF P,H COMBINATIONS SPECIFIED
60 C          - DIMENSION OF ARRAY PHWTS IS KPHWTS=4*NPH
61 C

```

```

62 C
63 C
64 C INPUTS
65 C
66 C     X(I)    COMPLEX MERGED REFERENCE SERIES ON WHICH THE PREDICTION
67 C           WEIGHTS ARE TO BE BASED
68 C
69 C     Y(I)    SERIES OF OBSERVATIONS FOR WHICH PREDICTION WEIGHTS ARE
70 C           TO BE FOUND (ZERO MEAN)
71 C
72 C     JP(I)   ARRAY OF P VALUES (X COMPONENT NUMBERS) TO BE USED
73 C           IN FORMING THE PREDICTION WEIGHTS
74 C
75 C     KP      DIMENSION OF ARRAY JP
76 C
77 C     H(I)    ARRAY OF H VALUES (X TIME LEADS) TO BE USED IN FORMING
78 C           THE PREDICTION WEIGHTS
79 C
80 C     KH      DIMENSION OF ARRAY H
81 C
82 C     NP(I)   NP(1) = 0
83 C           NP(I .GT. 1) IS THE NUMBER OF THE LAST TERM IN THE
84 C           (I-1)TH GROUP OF ARRAY JP
85 C
86 C     NH(I)   NH(1) = 0
87 C           NH(I .GT. 1) IS THE NUMBER OF THE LAST TERM IN THE
88 C           (I-1)TH GROUP OF ARRAY H
89 C
90 C     NPHGP1  DIMENSION OF ARRAY NP AND OF ARRAY NH
91 C
92 C     XXM1    AN ARRAY OF WORKING STORAGE NEEDED FOR MATRIX OPERATIONS
93 C
94 C -----COMMON /WAITER/
95 C
96 C     INISHL  INISHL .NE. 0 INDICATES THAT THIS IS THE INITIAL CALL
97 C           IN THIS COMPUTATION
98 C           INISHL .EQ. 0 INDICATES THAT THIS IS NOT THE INITIAL
99 C           CALL
100 C
101 C     IFINAL  IFINAL .NE. 0 INDICATES THAT THE CALL IS THE FINAL CALL
102 C           IN THIS COMPUTATION.
103 C           IFINAL = 0 INDICATES THAT THE CALL IS NOT THE FINAL CALL
104 C
105 C     SX      START TIME OF SERIES X
106 C
107 C     EX      END TIME OF SERIES X
108 C
109 C     SY      START TIME OF SERIES Y
110 C
111 C     EY      END TIME OF SERIES Y
112 C
113 C     D       UNIFORM TIME INCREMENT OF SERIES X AND SERIES Y
114 C
115 C     LIMP    NUMBER OF COMPLEX COMPONENTS MERGED IN SERIES X

```

```

116 C
117 C
118 C OUTPUTS
119 C
120 C   PHWTS(I) ARRAY CONSISTING OF P VALUE, H VALUE, COMPLEX PREDIC-
121 C   TION WEIGHT W(P,H) MERGED TOGETHER (I.E. 4 REAL COMPON-
122 C   ENTS MERGED)
123 C
124 C   KPHWTS   DIMENSION OF ARRAY PHWTS
125 C
126 C   WEIHTY   =0. IF NO ERRORS
127 C           =1. IF A P VALUE IS GREATER THAN THE NUMBER OF COM-
128 C           PONENTS IN THE X SERIES.
129 C           =2. IF A VALUE IN THE ARRAY NP EXCEEDS THE DIMENSION
130 C           OF ARRAY JP
131 C           =3. IF VALUES IN NP ARE NOT IN INCREASING ORDER
132 C           =4. IF A VALUE IN ARRAY NH EXCEEDS THE DIMENSION
133 C           OF ARRAY H
134 C           =5. IF VALUES IN NH ARE NOT IN INCREASING ORDER
135 C           =6. X SERIES STARTS TOO LATE TO ACCOMODATE MINIMUM LEAD
136 C           =7. SY IS NOT ONE OF THE TIMES OF SERIES X
137 C           =8. X SERIES ENDS TOO EARLY TO ACCOMODATE MAXIMUM LEAD
138 C           =9. IF NO P VALUES ARE GIVEN
139 C           =10. IF NO H VALUES ARE GIVEN
140 C           =11. IF NP(1) .LT. 0
141 C           =12. IF NH(1) .LT. 0
142 C           =13. IF THERE ARE NO P,H COMBINATIONS, THAT IS
143 C           NPHGP1 .LT. 2
144 C           =14. IF (TIME INTERVAL)/(END TIME-START TIME) IS
145 C           NEGATIVE FOR SERIES Y
146 C           =15. IF (TIME INTERVAL)/(END TIME-START TIME) IS
147 C           NEGATIVE FOR SERIES X
148 C           =16. IF MATRIX IS SINGULAR
149 C
150 C
151 C EXAMPLES
152 C
153 C
154 C PROGRAM FOLLOWS BELOW
155 C
156 C   COMMON /WAITER/ INISHL,IFINAL, SX,EX,SY,EY,D,LIMP
157 C   COMMON /WEIGHT/ JXY,NPH,NPH1,KY,TOT,SIGMA,JI,JK,JD
158 C   COMMON /WORKIN/ JH(125),JPH(125),XX1(250),YXV1(5^0)
159 C   DOUBLE PRECISION XXM,YXV
160 C   COMPLEX X
161 C   DIMENSION X(1),Y(1),JP(KP),H(KH),NP(NPHGP1),NH(NPHGP1),
162 C   1 XXM1(1),XXM(1),YXV(250),PPHG(125)   !!!
163 C
164 C--- (SEE @NOTE@ ABOVE FOR TRUE DIMENSIONS OF X,Y,XXM1,PHWTS)
165 C
166 C   EQUIVALENCE (YXV1(1),YXV(1)), (PPHG(1),JH(1))
167 C
168 C   DATA DELTO,DELO/0.8,0.8/
169 C

```

```

170      WEIHTY=0.
171      IF(SX.EQ.EX) GO TO 10
172      IF(D/(EX-SX).GT.0.) GO TO 10
173      WEIHTY=15.
174      RETURN
175  10  IF(SY.WE.EY) GO TO 20
176      KY=1
177      GO TO 40
178  20  DDEMSY=D/(EY-SY)
179      IF(DDEMSY.GT.0.) GO TO 30
180      WEIHTY=14.
181      RETURN
182  30  KY=INT(1./DDEMSY+1.5)
183  C      SKIP NEXT SECTION IF NOT INITIAL CALU
184  40  IF(INISHL.EQ.0) GO TO 100
185  c\////////////////////////////////////
186      IF(KP.LT.1) WEIHTY=9.
187      IF(KH.LT.1) WEIHTY=10.
188      IF(NP(1).LT.0) WEIHTY=11.
189      IF(NH(1).LT.0) WEIHTY=12.
190      IF(NPHGP1.LT.2) WEIHTY=13.
191      IF(WEIHTY.WE.0.) RETURN
192  c\////////////////////////////////////
193      DO 50 K=1,KP
194          IF(JP(K).GT.LIMP) WEIHTY=1.
195  50  CONTINUE
196  c\////////////////////////////////////
197      HMIN=H(1)
198      HMAX=H(1)
199      DO 60 K=1,KH
200          JH(K)=IROUND(H(K)/D)
201          IF(H(K).LT.HMIN) HMIN=H(K)
202          IF(H(K).GT.HMAX) HMAX=H(K)
203  60  CONTINUE
204  c\////////////////////////////////////
205      NPH=0
206  C      FOR EACH P,H COMBINATION, COMPUTE OFFSET JPH IN SERIES X
207      DO 70 IPHGP1=2,NPHGP1
208          NP1=NP(IPHGP1-1)+1
209          NP2=NP(IPHGP1)
210          IF(NP2.GT.KP) WEIHTY=2.
211          IF(NP2.LT.NP1) WEIHTY=3.
212          NH1=NH(IPHGP1-1)+1
213          NH2=NH(IPHGP1)
214          IF(NH2.GT.KH) WEIHTY=4.
215          IF(NH2.LT.NH1) WEIHTY=5.
216          DO 70 IP=NP1,NP2
217              JPIP=JP(IP)
218              DO 70 IH=NH1,NH2
219                  NPH=NPH+1
220  70  JPH(NPH)=JPIP+JH(IH)*LIMP
221  c\////////////////////////////////////
222      NPH1=2*NPH
223      KPHWTS=2*NPH1

```



```

224 c\////////////////////////////////
225 100 XY=SY-SI
226 IF(-XY-HMIN.GT.DELTO) WEIHTY=6.
227 XY=XY/D
228 JXY=IROUND(XY)
229 IF(ABS(XY-FLOAT(JXY)).GT.DELO) WEIHTY=7.
230 IF(EY+HMAX-EX.GT.DELTO) WEIHTY=8.
231 IF(WEIHTY.NE.0.) RETURN
232 JXY=JXY*LIMP
233 C FORM MATRIX M (UPPER TRIANGULAR) AND VECTOR V (BOTH IN
234 C DOUBLE PRECISION)
235 CALL MAVDUB(Y,X,XXM)
236 C RETURN IF NOT FINAL CALL
237 IF(IFINAL.EQ.0) RETURN
238 c\////////////////////////////////
239 C PACK M AND V IN SINGLE PRECISION FORM
240
241 IJM1=WPH1+1
242 IJM2=WPH
243
244 DO 120 JPH1=1,WPH1
245 DO 110 IPH1=1,JPH1
246 IJM1=IJM1+1
247 IJM2=IJM2+1
248 XXM1(IJM1)=SNGL(XXM(IJM2))/TOT
249 110 continue
250 YXV1(JPH1)=SNGL(YXV(JPH1))/TOT
251 120 continue
252
253 c\////////////////////////////////
254 SIGMA=SIGMA/TOT
255 C SOLVE EQUATION FOR VECTOR OF WEIGHTS W
256 IF(ISMI(XXM1).NE.0) GO TO 160
257 P=0.
258 I=0
259 IPHG=0
260 DO 150 IPHG=1,WPHGP1
261 IPHG=IPHG+1
262 NP1=NP(IPHG-1)+1
263 NP2=NP(IPHG)
264 NH1=NH(IPHG-1)+1
265 NH2=NH(IPHG)
266 PRINT 1000, IPHG,(H(IH), IH=NH1,NH2)
267 PRINT 1100
268 PG=0.
269 DO 140 IP=NP1,NP2
270 PPG=0.
271 I1=I+1
272 DO 130 IH=NH1,NH2
273 PHWTS(4*I+1)=FLOAT(JP(IP))
274 PHWTS(4*I+2)=H(IH)
275 PHWTS(4*I+3)=YXV1(2*I+201)
276 PHWTS(4*I+4)=YXV1(2*I+202)
277 I=I+1

```

```

278          PPHG(I)=YXV1(2*I-1)*YXV1(2*I+199)+YXV1(2*I)*YXV1(2*I+200)
279 130      PPG=PPG+PPHG(I)
280          PRINT 1200, JP(IP),PPG,(PPHG(IH), IH=I1,I)
281 140      PG=PG+PPG
282          PRINT 1300, PG,IPHG
283 150      P=P+PG
284
285          RESIDV=SIGMA-P
286          PRINT 1400, SIGMA,P,RESIDV
287          I=4*I
288          PRINT 1500, (PHWTS(K), K=1,I)
289          PRINT 1600
290          RETURN
291 160      WEIGHTY=16.
292          RETURN
293
294 c//////////
295
296 1000 FORMAT (40H0 PREDICTED VARIANCE MATRIX OF P,H GROUP,I3,3H IS/1H0,
297          15X,12HROW SUM      H,F11.4,7F12.4/(17X,8F12.4))
298 1100 FORMAT (3H P)
299 1200 FORMAT (1X,I2,2E13.4,7E12.4/(17X,8E12.4))
300 1300 FORMAT (5X,11H-----/4X,E12.4,42H IS TOTAL PREDICTED VARIANC
301          1E OF P,H GROUP,I3/)
302 1400 FORMAT (1H0,5X,21HRECORDED VARIANCE =,E14.6/6X,21HPREDICTED VARI
303          1ANCE =,E14.6/6X,21HRESIDUAL VARIANCE =,E14.6//)
304 1500 FORMAT (21H0 GENERATED SERIES IS/3H0 P,6X,1HH,8X,
305          + 26HREAL(WEIGHT) 1IMAG(WEIGHT)/(1X,F2.0,F12.4,2E14.4) )
306 1600 FORMAT (1H0//)
307
308          END
1          FUNCTION SPONTY(X, PHWTS,KPHWTS, JP, NP,NPGP1, YPRED,KY)
2 C
3 C
4 C TITLE - SPONTY = RESPONSE, TYDE
5 C      GENERATES TIDE SERIES FROM PREDICTION WEIGHTS
6 C
7 C
8 C          ---ABSTRACT---
9 C
10 C          SPONTY FORMS TIME SERIES Y(P,T) = SUM OVER H OF
11 C          CONJG(W(P,H))*X(P,T+H). W(P,H) REPRESENTS THE COMPLEX
12 C          WEIGHT ASSOCIATED WITH THE PTH COMPONENT OF THE REFERENCE
13 C          SERIES AT LEAD H. W(P,H) IS OBTAINED FROM INPUT SERIES
14 C          PHWTS CONSISTING OF P, H, W(P,H) MERGED (I.E. A 4
15 C          REAL COMPONENT SERIES). X(P,T) REPRESENTS THE PTH
16 C          COMPONENT OF THE COMPLEX REFERENCE SERIES X AT TIME T.
17 C          IF KOMPLX IS ZERO ONLY THE REAL PART OF Y(P,T) IS RE-
18 C          TAINED. THE P VLAUES TO BE USED ARE SPECIFIED BY ARRAY
19 C          JP. UNLESS JP(1) IS ZERO, THE ONLY P VAULES USED ARE
20 C          THOSE GIVEN BY JP(1), JP(2), ..., JP(N) WHERE
21 C          N = NP(NPGP1), AND THE OUTPUT SERIES YPRED(IPGP,T)
22 C          CONSISTS OF THE MERGED COMPONENT SUMS Y(JP(I),T)
23 C          + Y(JP(I+1),T) + ... + Y(JP(IP),T) WHERE I=NP(IPGP)+1

```

```

24 C      AND IP=NP(IPGP+1) AND IPGP=1,2,...,(NPGP1-1). IF JP(1)
25 C      IS ZERO THEN ALL P VALUES IN PHWTS ARE USED AND
26 C      YPRED(T) IS THE SINGLE (REAL OR COMPLEX) COMPONENT
27 C      SERIES CONSISTING OF THE SUM OF Y(P,T) OVER ALL P.
28 C
29 C
30 C      --STATISTICS--
31 C
32 C LANGUAGE - UCSD FORTRAN 63
33 C EQUIPMENT - NO SPECIFIED REQUIREMENTS
34 C STORAGE - 261 WORDS FOR THE PROGRAM + 1008 WORDS COMMON (480 OF
35 C      WHICH ARE DUMMY) + SPACE FOR ARRAYS X, PHWTS, JP,
36 C      NP, YPRED (I.E. 2*KX+KPHWTS+KP+NPGP1+K12*KY WORDS, WHERE
37 C      K12=1 OR 2 ACCORDING AS KOMPLX IS ZERO OR NON-ZERO)
38 C SPEED -
39 C AUTHOR - MARK WIMBUSH IGPP AUG 1970
40 C LAST MOD - MARK WIMBUSH NOVA APR 1972
41 C CATAGORIES -
42 C STATUS -
43 C LIBRARY ROUTINES USED - NONE
44 C SYSTEM ROUTINES USED - IROUND, XINTF, AIMAG, FLOATF
45 C
46 C
47 C      ----USAGE----
48 C
49 C SAMPLE CALL
50 C      J = SPONTY(X,PHWTS,KPHWTS,JP,NP,NPGP1, YPRED,KY)
51 C
52 C NOTE - DIMENSION OF REFERENCE SERIES X IS
53 C      LIMP*INT(1.5+(EX-SX)/D) (COMPLEX)
54 C      - DIMENSION OF COMPONENT NUMBER SERIES JP IS NP(NPGP1)
55 C      - DIMENSION OF PREDICTED SERIES YPRED IS
56 C      KY=INT(1.5+(EY-SY)/D) (YPRED CONSIDERED COMPLEX
57 C      UNLESS KOMPLX=0)
58 C
59 C
60 C INPUTS
61 C
62 C      X(I) COMPLEX MERGED REFERENCE SERIES ON WHICH THE PREDICTION
63 C      WEIGHTS ARE BASED
64 C
65 C      PHWTS(I) ARRAY CONSISTING OF P VALUE, H VALUE, COMPLEX PRE-
66 C      DICTIION WEIGHT W(P,H) MERGED TOGETHER (I.E. 4 REAL
67 C      COMPONENTS MERGED)
68 C
69 C      KPHWTS DIMENSION OF ARRAY PHWTS (SHOULD BE 4*NPH WHERE NPH
70 C      IS THE TOTAL NUMBER OF P,H COMBINATIONS IN PHWTS)
71 C
72 C      JP(I) ARRAY OF P VALUES (X COMPONENT NUMBERS) TO BE USED IN
73 C      FORMING THE PREDICTED TIDE
74 C
75 C      NP(I) NP(1) = 0
76 C      NP(I .GT. 1) IS THE NUMBER OF THE LAST TERM IN THE
77 C      (I-1)TH GROUP OF ARRAY JP

```

```

78 C
79 C      NPGP1   DIMENSION OF ARRAY NP
80 C
81 C ----COMMON /RESTOR/
82 C
83 C      IWISHL  IWISHL .NE. 0  INDICATES THAT THE CALL IS THE INITIAL
84 C                CALL IN THIS COMPUTATION
85 C                IWISHL .EQ. 0  INDICATES THAT THE CALL IS NOT THE
86 C                INITIAL CALL
87 C
88 C      SX      START TIME OF SERIES X
89 C
90 C      EX      END TIME OF SERIES X
91 C
92 C      SY      START TIME OF SERIES YPRED
93 C
94 C      EY      END TIME OF SERIES YPRED
95 C
96 C      D       UNIFORM TIME INCREMENT OF SERIES X AND SERIES YPRED
97 C
98 C      LIMP    NUMBER OF COMPLEX COMPONENTS MERGED IN SERIES X
99 C
100 C     KOMPLX  KOMPLX .NE. 0  INDICATES THAT THE COMPLEX PREDICTED
101 C                SERIES IS REQUIRED
102 C                KOMPLX .EQ. 0  INDICATES THAT ONLY THE REAL PART OF THE
103 C                PREDICTED SERIES IS TO BE RETAINED
104 C
105 C
106 C  OUTPUTS
107 C
108 C     YPRED(I) TIME SERIES OF TIDE PREDICTIONS, HAVING NPGP1-1 MERGED
109 C                (REAL OR COMPLEX) COMPONENTS IF JP(1) .NE. 0, OTHERWISE
110 C                HAVING ONE (REAL OR COMPLEX) COMPONENT
111 C
112 C     KY      DIMENSION OF SERIES YPRED (YPRED CONSIDERED COMPLEX
113 C                UNLESS KOMPLX=0)
114 C
115 C     SPONTY  =0. IF INPUT ITEMS HAVE VALID VALUES
116 C            =1. IF VALUES IN NP ARE NOT IN INCREASING ORDER
117 C            =2. IF THE POS GIVEN IN ARRAY JP DO NOT MATCH THE
118 C            POS GIVEN IN SERIES PHWTS
119 C            =3. X SERIES STARTS TOO LATE TO ACCOMODATE MINIMUM LEAD
120 C            =4. IF SY IS NOT ONE OF THE TIMES OF SERIES X
121 C            =5. X SERIES ENDS TOO EARLY TO ACCOMODATE MAXIMUM LEAD
122 C            =6. IF NP(1) .LT. 0
123 C            =7. IF THERE ARE NO P GROUPS, THAT IS NPGP1 .LT. 2
124 C            =8. IF KPHWTS INVALID (.LE.0 OR NOT A MULTIPLE OF 4)
125 C            =9. IF (TIME INTERVAL)/(END TIME - START TIME) IS
126 C            NEGATIVE FOR SERIES YPRED
127 C
128 C
129 C  EXAMPLES
130 C
131 C

```

```

132 C PROGRAM FOLLOWS BELOW
133 C
134 COMMON /RESTOR/ IWISHL,SI,EX,SY,EY,D,LIMP,KOMPLX
135 COMMON /WORKIN/ RSVP(500),JPOPHW(125),JPHW(125),JPH(125),MPH(125)
136 c//////////T//////////://////////
137 COMPLEX X
138 DIMENSION X(1),PHWTS(KPHWTS),JP(1),NP(NPGP1),YPRED(1)
139 C (SEE NOTE ABOVE FOR TRUE DIMENSIONS OF X,JP,YPRED)
140 EQUIVALENCE (JPIP,RSVP(1)),(JXY,RSVP(2)),(J,RSVP(3)),
141 1 (INC,RSVP(4)),(LP,RSVP(5)),(MPH1,RSVP(6)),(MPH2,RSVP(7)),
142 2 (NPGI,RSVP(8)),(NPGINT,RSVP(9)),(NPH,RSVP(10)),(NPHT,RSVP(11)),
143 3 (NPHW,RSVP(12)),(NP1,RSVP(13)),(NP2,RSVP(14)),(DDEMSY,RSVP(15)),
144 4 (DD1E3,RSVP(16)),(HMAX,RSVP(17)),(HMIN,RSVP(18)),(XY,RSVP(19))
145 DATA DELTO,DELO/0.8,0.8/
146
147 SPONTY=0.
148 IF(SY.NE.EY) GO TO 10
149 KY=1
150 GO TO 30
151 10 DDEMSY=D/(EY-SY)
152 IF(DDEMSY.GT.0.) GO TO 20
153 SPONTY=9.
154 RETURN
155 20 KY=INT(1./DDEMSY+1.5)
156 C SKIP NEXT SECTION IF NOT INITIAL CALL
157 30 IF(IWISHL.EQ.0) GO TO 100
158 IF(JP(1).NE.0) GO TO 34
159 NP(1)=0
160 NP(2)=1
161 NPGP1=2
162 34 IF(NP(1).LT.0) SPONTY=6.
163 IF(NPGP1.LT.2) SPONTY=7.
164 IF(MOD(KPHWTS-3,4)-1.NE.0) SPONTY=8.
165 IF(SPONTY.NE.0.) RETURN
166 NPHW=0
167 NPH=0
168 MPH(1)=0
169 HMIN=PHWTS(2)
170 HMAX=PHWTS(2)
171 DO 40 IPHWTS=1,KPHWTS,4
172 IF(PHWTS(IPHWTS+1).LT.HMIN) HMIN=PHWTS(IPHWTS+1)
173 IF(PHWTS(IPHWTS+1).GT.HMAX) HMAX=PHWTS(IPHWTS+1)
174 NPHW=NPHW+1
175 C--- (JPOPHW(I) IS THE IOTH P IN MERGED INPUT SERIES PHWTS)
176 40 JPOPHW(NPHW)=INT(PHWTS(IPHWTS))
177
178 C--- FOR EACH NEEDED P,H COMBINATION COMPUTE OFFSET JPH IN SERIES X
179
180 DO 80 IPGP1=2,NPGP1
181 NP1=NP(IPGP1-1)+1
182 NP2=NP(IPGP1)
183 IF(NP2.LT.NP1) SPONTY=1.
184 DO 70 IP=NP1,NP2
185 NPHT=NPH

```

```

186      JPIP=JP(IP)
187      DO 50 IPHW=1,NPHW
188          IF(JPOPHW(IPHW).NE.JPIP.AND.JP(1).NE.0) GO TO 50
189          NPH=NPH+1
190 C---      (JPHW(I) IS NUMBER IN THE MERGED INPUT SERIES PHWTS
191 C---      OF THE IOTH SELECTED P,H,W GROUP)
192          JPHW(NPH)=IPHW
193          JPH(NPH)=JPOPHW(IPHW)+IROUND(PHWTS(4*IPHW-2)/D)*LIMP
194      50    CONTINUE
195          IF(NPHT.NE.NPH) GO TO 70
196          SPONTY=2.
197      70    CONTINUE
198
199 C---      (MPH(I+1) IS THE SELECTION NUMBER OF THE LAST P,H,W
200 C---      GROUP ASSOCIATED WITH THE LAST P IN THE IOTH GROUP
201 C---      IN THE STATEMENT LIST (MPH(1)=0) )
202
203          MPH(IPGP1)=NPH
204      80    continue
205
206          INC=1
207          IF(KOMPLX.NE.0) INC=2
208          NPGI=(NPGP1-1)*INC
209      100   XY=SY-SX
210          IF(-XY-HMIN.GT.DELTO) SPONTY=3.
211          XY=XY/D
212          JXY=IROUND(XY)
213          IF(ABS(XY-FLOAT(JXY)).GT.DELO) SPONTY=4.
214          IF(EY+HMAX-EX.GT.DELTO) SPONTY=5.
215          IF(SPONTY.NE.0.) RETURN
216          JXY=JXY*LIMP
217          NPGINT=NPGI*KY
218 C        SET TO ZERO ARRAY YPRED
219
220          DO 110 IT=1,NPGINT
221              YPRED(IT)=0.
222      110   continue
223
224          LP=1
225
226 C---      COMPUTE PREDICTED SERIES AND STORE IN ARRAY YPRED IN MERGED
227 C---      FORM
228
229          DO 140 IT=1,KY
230              DO 130 IPGP1=2,NPGP1
231                  MPH1=MPH(IPGP1-1)+1
232                  MPH2=MPH(IPGP1)
233                  DO 120 IPH=MPH1,MPH2
234                      J=JXY+JPH(IPH)
235                      IPHW=JPHW(IPH)
236                      YPRED(LP)=YPRED(LP)
237      1      +PHWTS(4*IPHW-1)*REAL(X(J))+PHWTS(4*IPHW)*AIMAG(X(J))
238                      IF(KOMPLX.NE.0) YPRED(LP+1)=YPRED(LP+1)
239      1      +PHWTS(4*IPHW-1)*AIMAG(X(J))-PHWTS(4*IPHW)*REAL(X(J))

```

```

240 120 CONTINUE
241 130 LP=LP+INC
242 140 JXY=JXY+LIMP
243
244 RETURN
245
246 END
1 ! @FOR,IS TIDE.TADM,TIDE.TADM
2 FUNCTION TADM(PHW,LI, ORAY,LO, FS,DELF,CMPWT, KEEP)
3 DIMENSION PHW(LI),ORAY(LO)
4 DIMENSION FREQ(49)
5 REAL*8 KSYMB(50)
6 C
7 C DARWIN SYMBOLS AND FREQUENCIES STORED IN ORDER OF ASCENDING
8 C SIGNIFICANCE.
9 C
10 DATA KSYMB /2HS8, 2HS6, 2HS4, 2HS3, 4H2SM2, 2HS1, 3HMSF, 3HSSA,
11 1 5H2MSN8, 4HMSN6, 4H2SM6, 3HMK4, 3NSO3, 3HSK3, 4HMMS2, 3H2Q1,
12 2 3HP11, 6HSIGMA1, 3HR01, 3H001, 2HMF, 2HSA, 6H2(MS)8, 4H2MN6,
13 3 3HMN4, 4H2MK3, 3H2W2, 2HT2, 2HL2, 3HMU2, 2HM1, 2HJ1, 4H3MS8,
14 4 4H2NS6, 3HMS4, 3HMK3, 3HWU2, 2HQ1, 2HM6, 2HM6, 2HM4, 2HM3, 2HK2,
15 5 2HW2, 2HP1, 2H01, 2HS2, 2HK1, 2HM2, 1H /
16 C
17 DATA FREQ / 8.0, 6.0, 4.0, 3.0, 2.067726385, 1.0, .0677263854,
18 1 .0054758186, 7.760529198, 5.828255583, 5.932273615, 3.937749433,
19 2 2.929535705, 3.002737909, 1.828255583, .8569524129, .9945243121,
20 3 .8618093199, .8981009661, 1.075940113, .073202204, .0027379093,
21 4 7.86454722, 5.760529198, 3.828255583, 2.86180932, 1.859690322,
22 5 1.997262221, 1.968565261, 1.864547229, .9664462631, 1.039029556,
23 6 7.796820844, 5.864547229, 3.932273615, 2.935011524, 1.900838875,
24 7 .8932440591, 7.729094458, 5.796820844, 3.864547229, 2.898410422,
25 8 2.005475819, 1.895981968, .9972620907, .9295357053, 2.0,
26 9 1.002737909, 1.932273615/
27 C
28 1 FORMAT(F6.4)
29 2 FORMAT(1H0,F10.3,F10.4,F11.6,F13.6,F12.6,F13.6,F10.3,2(5X,A7,
30 1 8X))
31 3 FORMAT(1H0,8X,17HF R E Q U E N C Y,9X,1H*,12X,19H A D M I T T A N C
32 1 E,14X,1H*,8X,21H T I D A L L I N E S/
33 2 44X,28H$NAMOBS LEADS $NAMREF BY DEG/
34 3 6X,37HCPY CPM CPD * REAL,8X,67HIMAG * ANPLI
35 4TUDE DEG * MOST SIGNIFICANT * MOST CENTRAL/
36 5 35X,1H*,22X,1H*,22X,1H*,13X,8HCPD *,12X,3HCPD)
37 4 FORMAT(1H0)
38 C
39 C LO IS LENGTH OF ORAY
40 C LI IS LENGTH OF PHW
41 C THE LENGTH OF PHW SHOULD BE A MULTIPLE OF 4
42 TADM = 1.
43 IF(MOD(LI,4) .NE. 0) RETURN
44 C
45 PRINT 3
46 PI = -3.1415926536
47 F = FS

```

```

48 C
49 DO 3000 N1=1,LO,2
50 N2=N1 + 1
51 OREAL = 0.
52 OIMAG = 0.
53 C
54 DO 2000 K1=1,LI,4
55 K2=K1 + 1
56 K3 = K2 + 1
57 K4 = K3 + 1
58 IF(PHW(K1) .NE. CMPNT) GO TO 2000
59 ALPH = PHW(K2) * PI * F / 12.
60 CF = COS(ALPH)
61 SF = SIN(ALPH)
62 OREAL = OREAL + PHW(K3) * CF + PHW(K4) * SF
63 OIMAG = OIMAG + PHW(K4) * CF - PHW(K3) * SF
64 2000 CONTINUE
65 C
66 C CPM = 27.321582 TROPICAL MONTH
67 CPM = F * 27.321582
68 C CPY = 365.25 JULIAN YEAR
69 CPY = F * 365.25
70 C NOTE THAT A TROPICAL YEAR IS 365.24219879 FOR THE YEAR 1900 AND
71 C SHOULD HAVE .00000600 SUBTRACTED FOR EACH CENTURY AFTER 1900.
72 R = SQRT(OREAL**2 + OIMAG**2)
73 PSI = ATAN2(OIMAG,OREAL) * 57.29578
74 FMAX = F + DELF/2.
75 FMIN = FMAX - DELF
76 C
77 C PICK THE DARWIN SYMBOL AND FREQUENCY OF BOTH THE MOST CENTRAL
78 C LINE AND THE MOST SIGNIFICANT LINE CONTAINED IN ANY BAND
79 C F PLUS OR MINUS DELF. SIGNIFICANCE IS TAKEN TO BE THAT IMPLIED
80 C BY THE ORDERING OF THE LINES IN THE HARMONIC CONSTANTS TABLES
81 C OF THE INTERNATIONAL HYDROGRAPHIC BUREAU (MONACO), WITH
82 C SPECIES INTERSPERSED.
83 C
84 C SET DIFF = LARGE NUMBER
85 DIFF =10.**35
86 JNEAR = 50
87 JSIG=50
88 C
89 DO 2200 J=1,49
90 IF(FREQ(J) .GT. FMAX) GO TO 2200
91 IF(FREQ(J) .LT. FMIN) GO TO 2200
92 JSIG = J
93 T = ABS(F - FREQ(J))
94 IF(T .GT. DIFF) GO TO 2200
95 DIFF = T
96 JNEAR = J
97 2200 CONTINUE
98 C
99 IF(JNEAR .EQ. 50) GO TO 3400
100 PRINT 22, CPY,CPM,F,OREAL,OIMAG,R,PSI,KSYMB(JSIG),FREQ(JSIG),
101 1 KSYMB(JNEAR),FREQ(JNEAR)

```



```

102     22 FORMAT(1H0,F10.3,F10.4,F11.6,F13.6,F12.6,F13.6,F10.3,
103     1 2(5X,A7,F6.4,2X))
104     2400 IF(KEEP .EQ. 0) GO TO 2500
105         ORAY(N1) = OREAL
106         ORAY(N2) = OIMAG
107     2500 F = F + DELF
108     3000 CONTINUE
109 C
110         TADM = 0.
111         PRINT 4
112         RETURN
113     3400 PRINT 2,CPY,CPM,F,OREAL ,OIMAG ,R,PSI,KSYMB(JSIG),KSYMB(JWEAR)
114         GO TO 2400
115     END
1     SUBROUTINE HG(ORAY,LO,nCONST,NMC,MORDER,NTYPE)
2 C
3 C     TITLE - HG
4 C     COMPUTES LINE AMPLITUDES IN CENTIMETERS AND GREENWICH
5 C     PHASE FROM RESPONSE ADMITTANCES
6 C     AUTHOR- MARK WIMBUSH     NOVA 1975
7 C
8     DIMENSION HPOTO2(2),HPOT(2,2,2),HPOTX3(2,3,2),ORAY(LO),
9     &          NCONST(NMC)
10    DATA HPOTO2/3.1,6.663/
11    DATA HPOT/26.221,36.878,5.02,12.203,63.192,7.996,12.099,29.4/
12    DATA HPOTX3/1.3871,0.05969,0.2314,0.22144,0.55741,
13    &          0.048,0.399,0.146,0.389,0.359,0.210,0.765/
14
15    print 120
16
17    DO N = 1,NMC
18        MC = NCONST(N)
19        OREAL = ORAY(2*MC-1)
20        OIMAG = ORAY(2*MC)
21        R = SQRT(OREAL**2+OIMAG**2)
22        PSI = ATAN2(OIMAG,OREAL)*57.29578
23        G = AMOD(180.0*FLOAT(MORDER)-PSI,360.0)
24
25        IF (NTYPE.EQ.10) then
26            J = 1
27            IF (NCONST(1).EQ.1) J = 2
28            H = R*HPOT(N,J,MORDER)
29        else IF (NTYPE.EQ.-10) then
30            H = R*HPOTO2(N)
31        else
32            IF (NTYPE.GT.0) then
33                I = 2
34                J = NTYPE
35            else
36                I = 1
37                J = 1-NTYPE
38            endif
39            H = R*HPOTX3(N,J,I)
40        endif

```

```
41
42     print 160,NC,H,G
43     enddo
44
45     -
46     RETURN
47
48 120  FORMAT('0',18X,'H',20X,'G')
49 160  FORMAT('0',1X,I3,F20.5,F20.3)
50
51     END
```

3.9 FILTER_NAMES.M

```

1  %*****
2  % filter_names.m
3  %
4  % filters the ies records specified in the file names.m
5  % 24-Jan-1990
6  %*****
7  % enter the file names, bints and filter coefficients from quasi-control
8  % file names.m.
9  %
10 % z      = array of seacor file prefixes
11 % bints  = vector containing B-intercepts
12 % b,a    = butterworth filter coefficients, as in signal processing tool box
13 %                documentation, or 'help butter' in Matlab
14 %
15 echo off
16 names                    % get z,bints,b, and a
17 for i=1:length(bints)    % loop through all files
18
19 eval(['load ',z(i,:),'.seacor']) % load file. Eval(t) is a text macro
20 % facility. It causes text string in
21 % t to be interpreted as a matlab
22 % command or expression. 'help eval'
23 % or 3-39 in the manual.
24
25 eval(['y = ',z(i,:),'(:,2);']) % assign travel times to vector y
26 eval(['t = ',z(i,:),'(:,4);']) % assign time to vector t
27
28 k = rampf(y);             % remove ramp (line from first to last
29 y = y - k;               %                point) not a Matlab m-file
30
31 y = filter(b,a,y);       % filter forward. See 3-47 in Matlab
32 % manual or 'help filter'
33
34 y = flipx(filter(b,a,flipx(y))); % filter backwards. flipx flips a
35 % matrix about the x axis. here it
36 % is used to reverse the order of a
37 % column vector (and will do nothing to a
38 % row vector). also see flipy.
39
40 y = y + k;               % return ramp
41
42
43 y = y(41:length(y)-40); % remove regions with possible
44 t = t(41:length(t)-40); % transient ringing
45
46 [sy,st] = subsample(y,t); % subsample at even 6 hourly
47 % increments. not a Matlab m-file
48
49 sy=-19800*sy + bints(i)*ones(length(sy),1);
50 % calibrate to Z_12 depth. ones(n,m)
51 % creates a nxm matrix of ones. 3-89
52

```

```
53
54 q = [st,sy]; % save to mat file
55 eval(['save ',z(i,:),'.z12star q /ascii'])
56 eval(['clear 'z(i,_)'])
57 end % end for loop
58
```

-

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.
- Rosby, 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 DC207:

```
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.

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

```

6 C***      string is used in name.
7 C*** headr  = used for the header card of the control file
8 C*** ...
9 C***
10 character*60 name,filename,file
11 character*60 headr
12 character*3 cruise
13 character*2 site,instr,yesno(2)
14 character*1 type
15 integer recs,yearspan,status,lib$spawn,size,str$trim
16 logical exist
17
18 C****
19 C**** Get the file name (name) of file containing all the pertinent
20 C**** information to make the control file. This consists of:
21 C**** The instrument serial number, site, type of instrument (pressure
22 C**** or not), the total number of records, the length of the record,
23 C**** the number of years spanned by the record, whether or no the year
24 C**** was a leap one, ditto, the cruise number
25 C****
26 C**** fortran expressed as instr, site, type, recs, yearspan, yesno(2),
27 C**** cruise.
28 C****
29 C****
30 C**** The file has a header. A row of stars indicates that the next line is
31 C**** data containing the above entries.
32 C****
33 C****
34
35 2 write(5,*) 'input file name:'
36 read(5,fmt='(Q,A)') name_size, name(1:name_size)
37 inquire(file=name(1:name_size), exist=exist)
38
39 if (exist) then
40 open(unit=19,file=name(1:name_size),form='formatted',status='old')
41 else
42 type*, 'file not found.'
43 goto 2
44 endif
45
46 1 read(19,fmt='(Q,A)') name_size,name(1:name_size)
47 if (name(1:1).ne.'*') goto 1
48
49 3 read(19,100,end=200)
50 / instr,site,type,recs,yearspan,yesno,cruise
51 100 format(a2,x,a2,x,a1,x,i5,x,i1,x,a2,x,a2,x,a3)
52
53 C*****
54 C***** create file name, open it, and write appropriate cards for
55 C***** the given control file flag.
56 C*****
57 filename='[cruise.seacor.ctrl]'
58 file='ies90'
59 fsize=5

```

```

60 status = str$trim (filename,filename,size)
61 if (.not.status) call lib$signal(%val(status))
62 if ((type.ne.' ').or.(type.eq.' ')) then
63 file=type//ies90'
64 fsize=6
65 filename=filename(1:size)//type
66 size=size+1
67 endif
68 c filename = filename(1:size)//ies90'//site//'_ '//cruise//'.ctrl'
69 filename = filename(1:size)//ies90'//site//'.ctrl'
70 c file=file(1:fsize)//site//'_ '//cruise
71 file=file(1:fsize)//site
72 open(unit=20,file=filename,form='formatted',status='new')
73 headr='''//site// SNO'// instr// ' '//type// ' '//
74      / EN'//cruise//''''
75 write(20,103) headr,recs,years,span,yesno(1),yesno(2)
76 if (((ichar(site(1:1)).ge.97).and.(ichar(site(1:1)).le.99)).or.
77      / ((ichar(site(1:1)).ge.65).and.(ichar(site(1:1)).le.67))) then
78 write(20,201) 1
79 else
80 write(20,201) 3
81 endif
82
83 201 format(x,'$CARD3 region= ',i1 ,' $end')
84
85 202 close(unit=20)
86
87 C****
88 C**** Here name is used to generate a string that passes a command
89 C**** procedure call and the necessary parameter to the LIB$SPAWN
90 C**** routine. LIB$SPAWN allows the execution of the DCL procedure
91 C**** from within this program.
92 C****
93 C**** go_seacor simply executes seacor89.com and imprints the log file
94 C****
95 c name=@go_seacor '//file
96 name=@seacor90 '//file
97 status=lib$spawn(name)
98 C****
99 C**** get next instrument
100 C****
101 goto 3
102 200 close(unit=19)
103
104 103 format(' $CARD1'// HEADR='a60'// $END'// $CARD2
105          / NPTS=',i6,', NOYRS=',i1,', FRSTYR= ' ',a2,' ', SCNDYR
106          / =' ',a2,' ' $END')
107 end
108

```

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Distribution for public release; Distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) University of Rhode Island Graduate School of Oceanography		7a. NAME OF MONITORING ORGANIZATION	
6a. NAME OF PERFORMING ORGANIZATION Univ. of Rhode Island Grad. School of Ocean.	6b. OFFICE SYMBOL (if applicable) 1122 PO	7b. ADDRESS (City, State, and ZIP Code)	
5c. ADDRESS (City, State, and ZIP Code) South Ferry Road Narragansett, RI 02882		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-90-J-1568, N00014-90-J-1548, OCE87-17144-90	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Office of Naval Research National Science Foundation	8b. OFFICE SYMBOL (if applicable)	10. SOURCE OF FUNDING NUMBERS	
8c. ADDRESS (City, State, and ZIP Code) 800 N. Quincy St., Arlington, VA 22217 1800 G. Street, NW, Washington, DC 20550		PROGRAM ELEMENT NO.	PROJECT NO.
11. TITLE (Include Security Classification) Inverted Echo Sounder Data Processing Report		TASK NO.	WORK UNIT ACCESSION NO.
12. PERSONAL AUTHOR(S) Erik Fields, Karen Tracey and D. Randolph Watts			
13a. TYPE OF REPORT summary	13b. TIME COVERED FROM 1987 TO 1991	14. DATE OF REPORT (Year, Month, Day) May 1991	15. PAGE COUNT 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)			
<p>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.</p>			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION	
22a. NAME OF RESPONSIBLE INDIVIDUAL D. Randolph Watts		22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL