

AD-A121 498

DOCUMENTATION OF SOFTWARE FOR THE NEIL BROWN INSTRUMENT
SYSTEMS/NORDA VEL..(U) NAVAL OCEAN RESEARCH AND
DEVELOPMENT ACTIVITY NSTL STATION MS..

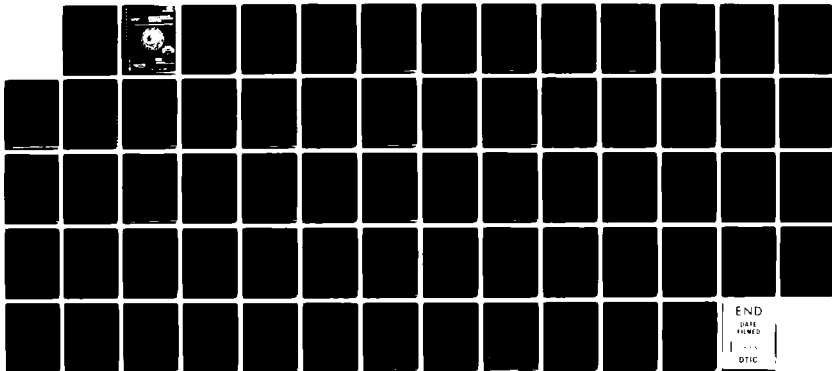
//

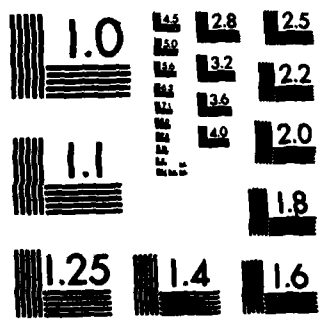
UNCLASSIFIED

K D SAUNDERS ET AL. JUL 82 NORDA-TN-155

F/G 8/10

NL





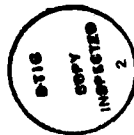
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A121498

ABSTRACT

Software developed for processing Neil Brown Instrument System/NORDA profiler data is documented in this report. The software includes programs for translating the profiler data from original NBIS format to engineering units in UNIVAC/NAVOCEANO FEB files and for editing and correcting the data subsequently. This report provides complete descriptions of the programs as well as operating information.

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	



ACKNOWLEDGMENTS

This work was sponsored by the NORDA Code 500 Ocean Measurements Program, Dr. Rudy Hollman, Program Manager.

CONTENTS

I. Introduction	1
II. Summary of Data Collection and Processing	1
III. Data Structures	2
IV. References	6
Appendix A: An estimator for vertical instrument velocity	7
Appendix B: Profiler software	9
Appendix C: Computer listings	15

DOCUMENTATION OF SOFTWARE FOR THE
NEIL BROWN INSTRUMENT SYSTEMS/NORDA VELOCITY/CTD PROFILER

I. INTRODUCTION

This report constitutes both documentation and user manual for software associated with the Neil Brown Instrument Systems/NORDA velocity/CTD profilers. As the software is in a constant state of development, this report is presented in a loose-leaf format, allowing for replacement of outdated software, correction of existing software and addition of new software, and operation instructions.

The initial software was written in HPL for the Hewlett-Packard 9825A calculator by Kim Saunders and Henry Perkins. The basic data collection and display programs were subsequently modified and expanded by James Vega of Computer Sciences Corporation. The 9825A is too small and slow for the data processing envisioned, requiring the use of a large mainframe or super-mini computer.

Vega wrote a set of three programs for the translation of profiler data to engineering units in UNIVAC 1108/NAVO/NORDA Fast Easy Binary (FEB) file format, the conversion of these data to orthogonal, geomagnetic coordinates, and the graphic display of these data. The first two of these programs were subsequently corrected and modified by Saunders.

Fred Hamrick of Computer Sciences Corporation wrote two programs to compute the vertical instrument velocity and correct the vertical velocity. The algorithms for these programs were developed by Saunders and Perkins, first testing in rough form on the HP 9825A.

After the GYRE cruise to the equatorial Atlantic in November-December 1981, it was found that a serious over-ranging problem occurred during occasionally strong down-swings of the instrument. The algorithm and program to approximately correct these errors were developed and implemented by Saunders.

These programs constitute, at present (April 1982), the existing software developed specifically for the profiler. NORDA Code 331 has, however, a growing suite of utility programs for the processing of FEB files. The documentation for these programs is not included in this report, but will constitute a separate report.

A DEC VAX 11/750 super-mini computer is in procurement for the sea-going data processing of the profiler data. New programs to appear in this report will soon be available and will include:

- real-time profiler data acquisition,
- acoustic navigation collection,
- real-time filtering and correction,
- real-time display.

II. SUMMARY OF DATA COLLECTION AND PROCESSING

The Neil Brown Instrument System/Naval Ocean Research and Development Activity 3-Axis Velocity/CTD profiler system was designed for the study of upper ocean mixing and variability. It is capable of measuring three components of velocity, acceleration, and magnetic field, as well as conductivity, temperature and pressure. This system and the initial phases of testing are documented in Perkins et al. (1980) and Saunders et al. (1981).

The profiler system consists of an underwater unit and a deck unit. The data are sensed by the underwater unit, digitized, and then transmitted to the surface via an audio FSK code over a single conductor (sea water return) cable. The signal is led through a winch with slip-rings to the deck unit, where the data are reformatted. There are three signal outputs from the deck unit:

- an audio output for backup on audio tape,
- an IEEE 488 parallel bus and,
- an output for a digital tape recorder (either a Kennedy or Digi-Data).

When the initial test of the profiler was conducted, the physical oceanography branch did not own a 9-track tape drive that could be directly connected to the third output. The branch owned an interface/buffer/recorder system that was compatible with the IEEE 488 bus. The initial data collection program was therefore designed to transfer data directly from the deck unit to the interface/buffer, occasionally breaking into the bus to obtain "snapshots" of the data being recorded. The present version of the data collection program "DATALOGGER" incorporates this design. However, it was recently discovered that because the interface/buffer is only a singly buffered system, about 15% of the data are lost when the buffer is emptied during recording. The solution to this problem is to record the data directly to magnetic tape via the third output.

The data collection program is also designed to produce quasi-realtime plots of the data on one or two printer-plotters and to display numeric data on a Hazeltine 1420 CRT terminal at 9600 Baud.

Once the data have been recorded on digital magnetic tape, they are processed on the UNIVAC 1108 at NAVOCEANO (this will be changed in the near future to a DEC VAX 11/750). The program which translates the data from the NBIS format to engineering units is "TRANSCRIBE." This program will handle data written directly from the deck unit to magnetic tape or tape generated by the IEEE 488-interface/buffer system.

The data should then be plotted and if there appears to be any evidence of over-ranging, the "PREFIX" program should be run. This program makes a close approximation to the velocities when over-ranging is encountered.

Once this is done, either "VFIX1" or "VFIX1-S" may be applied to the data. These programs attempt to correct the vertical velocity by computing and subtracting the instrument velocity from the observed vertical velocity. "VFIX1" accomplishes this by integrating the vertical acceleration, while "VFIX1-S" differentiates the pressure to obtain the instrument's vertical motion.

Two utility programs are also included. "UNORTHOG" is used mainly for testing and debugging purposes for looking at the velocity data in the original acoustic (non-orthogonal) components. "TSPLOT2" is a general plot package to plot any of the variables in the FEB files against either cycle number or time.

III. DATA STRUCTURES

Two primary structures are involved in profiler data processing: NBIS raw data format and profiler FEB files. The NBIS format is used only during the data collection phase. This is a highly packed format prior to conversion to engineering units. The FEB file structure is the standard file format used in the Physical Oceanography Branches at NORDA and NAVOCEANO (Hallock, 1981). The details of these formats are

given below for reference. The NBIS format is given in Figure 1 (from the NBIS profiler manual, with permission).

The FEB file variables, after the transcription phase are, in order of position in the data array:

No.	Name	Variable (units)
1	PRESS	pressure (decibars)
2	STEMP	slow response temperature (deg C)
3	COND	conductivity (mmho)
4	FTEMP	fast response temperature (deg C)
5	VLOC11	velocity component 1 (cm/sec)
6	VLOC12	velocity component 2 (cm/sec)
7	VLOC13	velocity component 3 (cm/sec)
8	MAG11	magnetic component 1
9	MAG12	magnetic component 2
10	MAG13	magnetic component 3
11	ACCI1	acceleration comp. 1
12	ACCI2	acceleration comp. 2
13	ACCI3	acceleration comp. 3
14	TIME	reference time (dec. days)
15	RELSEC	relative time (sec)

Further processing programs occasionally will change, correct, or replace these variables. For instance, after using PREFIX, variable 14 is replaced with an estimate of the instrument's vertical velocity, determined from the pressure time derivative. It is also possible to extend the number of variables in the FEB file structure, and at present the programs VFIX1 and VFIX1-S extend the number of variables to 17.

The header blocks for the PROFILER FEB files are defined as follows:

ADOC	(1) - (31): available for alphanumeric information.
FDOC	(1) - sample interval time
	(2) -
	(3) - start latitude (dec. deg.)
	(4) - start longitude (dec. deg.)
	(5) - time of start fix (dec. days)
	(6) - end latitude
	(7) - end longitude
	(8) - time of end fix
	(9) - maximum pressure of sensor
	(10) - cast start time (dec. days)
	(11) -
	(12) -
	(13) - cast end time
	(14) -
	(15) -
	(16) - magnetic variation
	(17) - magnetic dip
	(18) - ship's speed (kt)
	(19) - ship's heading
	(20) -
	(21) - dry bulb temp (deg. C)
	(22) - wet bulb temp (deg. C)

- (23) - surface temp (deg. C)
- (24) - barometric pressure (mb)
- (25) - wind speed (nm)
- (26) - wind direction (compass)
- (27) - significant wave height (ft)
- (28) -
- (29) -
- (30) -
- (31) - start time of profile (day)
- (32) - " " " " (hour)
- (33) - " " " " (min)
- (34) - " " " " (sec)
- (35) -
- (36) - end " " " (day)
- (37) - " " " " (hour)
- (38) - " " " " (min)
- (39) - " " " " (sec)
- (40) -

- IDOC
- (1) - end of profile flag
 - (2) - cruise number
 - (3) - station number (id format)
 - (4) - relative segment number
 - (5) - series sequence number
 - (6) - absolute no. of 1st seg. in series
 - (7) -
 - (8) -
 - (9) -
 - (10) - input tape no.
 - (11) - year of cast
 - (12) -
 - (13) - no. of bad cycles
 - (14) - no. of profiles in cast
 - (15) -
 - (16) -
 - (17) -
 - (18) -
 - (19) -
 - (20) -

Figure 1. NBIS Raw Data Format

BYTE	PARAMETER	DISPLAY UNITS	LS BIT WEIGHT	FORMAT*
1	Frame Sync.	240 or 015	---	---
2	Pressure LSB	see calibration	see calibration	AC
3	Pressure MSB			
4	Temperature LSB	degree celcius	0.5 m deg. C	AC
5	Temperature MSB			
6	Conductivity LSB	mmho	0.001 mmho	AC
7	Conductivity MSB			
8	Fast Temp. LSB	degree celcius	0.5 m deg. C	AC
9	Fast Temp. MSB			
10	AC Signs	part of temp. and pressure	---	SIGNS
11	Velocity X LSB	16383=1 m/sec	1/16383 m/sec	DC
12	Velocity X MSB			
13	Velocity Y LSB	16383=1 m/sec	1/16383 m/sec	DC
14	Velocity Y MSB			
15	Velocity Z LSB	16383=1 m/sec	1/16383 m/sec	DC
16	Velocity Z MSB			
17	Compass X LSB	ratio only	---	DC
18	Compass X MSB			
19	Compass Y LSB	ratio only	---	DC
20	Compass Y MSB			
21	Compass Z LSB	ratio only	---	DC
22	Compass Z MSB			
23	Acceleration X LSB	g's x 1000	0.001 g	DC
24	Acceleration x MSB			
25	Acceleration Y LSB	g's x 1000	0.001 g	DC
26	Acceleration Y MSB			
27	Acceleration Z LSB	g's x 1000	0.001 g	DC
28	Acceleration Z MSB			
29	Spare-0 LSB**	--	--	DC
30	Spare-0 MSB**			
31	Spare-1 LSB**	--	--	DC
32	Spare-1 MSB**			
33	TOD-0 msec	--	m sec x 10, msec x 1	BCD
34	TOD-1 sec/msec	--	sec x 1, msec x 100	BCD
35	TOD-2 min/sec	--	min x 1, sec x 10	BCD
36	TOD-3 hr/min	--	hr x 1, min x 10	BCD
37	TOD-4 day/hr	--	day x 1, hr x 10	BCD
38	TOD-5 day	--	day x 100, day x 10	BCD

- *NOTE
1. AC Format
LSB = 128, 64, 32, 16, 8, 4, 2, 1
MSB = 32768, 16384, 8192, 4096, 2048, 1024, 512, 256
 2. SIGNS Format
1, 1, 1, 1, 1, Fast Temp., Temp., Pressure
where 1 = negative, 0 = positive
 3. DC Format
LSB = 32, 16, 8, 4, 2, 1, 0, SIGN
MSB = 8192, 4096, 2048, 1024, 512, 256, 128, 64
 4. BCD Format
8, 4, 2, 1, 8, 4, 2, 1

** All bits in spare Bytes are set to "1"s.

Figure 1. NBIS Raw Data Format

IV. REFERENCES

Hallock, Z. R. (1980). The Fast and Easy Binary (FEB) File. NAVOCEANO TN 7210-12-80.

Perkins, H. T., K. D. Saunders, G. Appell, and T. Mero (1980). Design and Initial Testing of a Three Axis Acoustic Current Meter. OCEANS 80 Conference Record (IEEE Pub. No. 80CH1572-7), 319-322.

Saunders, K. D., H. T. Perkins, L. Banchemo, S. Sova, and J. J. Vega (1981). Sea Trials of a Lowered Three Axis Current Meter. OCEANS 81 Conference Record (IEEE Pub. No. 81CH1685-7), 245-249.

APPENDIX A: AN ESTIMATOR FOR VERTICAL INSTRUMENT VELOCITY

Let w_0 and a_0 represent observed vertical components of velocity and acceleration and let p_0 be the observed pressure. Suppose the data to be given over the time interval t_1, t_2

Define the instantaneous vertical velocity of the instrument as

$$w_1(t) = \int_{t_1}^t [\alpha a_0(t) - \bar{g} - \gamma] dt + \beta \quad (A-1)$$

NOTE: In the program, g was set equal to zero and a correction, $ACORR=9.99$, was applied to all measured accelerations.

where g is the local gravitational acceleration, α and γ are corrections to the observed acceleration and β is the instrument velocity at $t=t_1$. The values of α , β , and γ are to be determined in an optimal way, as described below. Nominally, $\alpha = 1$ and $\gamma = 0$.

Further define the mean vertical velocity of the instrument during t_1, t_2 on the basis of the corresponding pressure change:

$$\bar{w}_1 = \frac{1}{\bar{\rho}g} [p_0(t_1) - p_0(t_2)] / (t_2 - t_1) \quad (A-2)$$

Where $\bar{\rho}$ is the mean density of the water.

The quantities α, β, γ are determined by the conditions

$$\int_{t_1}^{t_2} (w_1 - w_0)^2 dt = \text{minimum} \quad (A-3)$$

and

$$\int_{t_1}^{t_2} (w_1 - \bar{w}) dt = 0 \quad (A-4)$$

That is, w_1 is required to resemble w_0 as much as possible and also correspond to the known mean instrument speed.

Formally, the two equations above constitute a linear least squares problem with a side condition. This may be reformulated by the method of Lagrange:

$$\int_{t_1}^{t_2} [(w_1 - w_0)^2 + \lambda(w_1 - \bar{w})] dt = \text{minimum} \quad (A-5)$$

Where λ is a Lagrange multiplier, the value of which is also to be determined. Differentiating this expression with respect to α, β, γ and λ in turn and equating each of the results to zero, as is required to minimize the expression, results in a linear system of 4 equations for $\alpha, \beta, \gamma, \lambda$. These have the form

$$AX = B \quad (A-6)$$

$$\text{Where } A = (A_{ij}) = \left(\int_{t_1}^{t_2} a_{ij}(t) dt \right) \quad (A-7)$$

$$X = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \lambda \end{pmatrix} \quad (A-8)$$

$$B = (B_j) = \left(\int_{t_1}^{t_2} b_j(t) dt \right) \quad (A-9)$$

NOTE: In the program, the last quantity in array B, $\int \bar{w}$, was set equal to the average velocity determined from the total pressure change multiplied by the total time.

The quantities a_{ij} , b_j can be shown to be

$$(a_{ij}) = \begin{pmatrix} f_1^2 & f_1 & -f_1 f_2 & \frac{1}{2} f_1 \\ f_1 & 1 & -f_2 & \frac{1}{2} \\ f_1 f_2 & f_2 & -f_2 & \frac{1}{2} f_2 \\ f_1 & 1 & -f_2 & 0 \end{pmatrix} \quad (A-10)$$

$$(b_j) = \begin{pmatrix} f_1(w_0 + g f_2) \\ w_0 + g f_2 \\ f_2(w_0 + g f_2) \\ \bar{w} + g f_2 \end{pmatrix} \quad (A-11)$$

$$\text{where } f_1 = f_1(t) = \int_{t_1}^{t_2} a_0(t) dt \quad (A-12)$$

$$f_2 = f_2(t) = \int_{t_1}^{t_2} dt = t - t_1 \quad (A-13)$$

Note that even though (a_{ij}) is singular, since the first three rows are multiples of each other, (A_{ij}) is not. Hence, it can be inverted, equations A-6 solved for α, β, γ and the result used in A-1 to find an optimal estimate of the instrument velocity.

APPENDIX B: PROFILER SOFTWARE

PROGRAM: DATALOGGER

PURPOSE: To log data and maintain a real time display of certain variables when the profiler is collecting data at sea. Data are logged by transferring the data stream directly to an IDEAS IEEE 488/CIPHER DATA Interface/buffer. The variables are displayed on a Hazeltine 1420 CRT terminal and plotted on two HP 7242 printer/plotters.

MACHINE: HEWLETT-PACKARD 9825A

LANGUAGE: HPL

AUTHOR: Kim David Saunders, Henry T. Perkins and James J. Vega

FILE LOCATIONS: Tape V002, File 4

INPUT: The program solicits the following:
Cast Number,
Latitude in degrees decimal,
Longitude in degrees decimal.
These are used only for display on the Hazeltine terminal. During normal operation of the program, different display options may be invoked by pressing the f0 function key. The program will then solicit the type of display option desired and the unit to which that type of display is to be directed. The display options are listed after the program listing.

OUTPUT: The output consists of alphanumeric listings of data on the Hazeltine screen and plots of data on either or both printer plotters.

ADDITIONAL INFORMATION:

PROGRAM: TRANSCRIBE

PURPOSE: To convert profiler data from raw form on the original data tapes (or condensed copies) into engineering units with the velocity expressed in orthogonal instrument coordinates. The converted data are stored in FEB file format for ease in further processing.

MACHINE: UNIVAC 1108

LANGUAGE: FORTRAN V

AUTHORS: James J. Vega, corrected and modified by Kim David Saunders

FILE LOCATIONS: Absolute, Relocatable, and Symbolic Elements - VEGA*LIB.TRANSCRIBE

INPUT: The input parameters are solicited by the program for use in an interactive environment.
The input raw data must be attached to logical unit 10, for example by a series of statements such as:

```
asg,tsj 10,u9s,<<tapeno.>>  
move 10,<<nfiles-1>>
```

The solicited input parameters are summarized below for use in a batch environment. All input is in free format.

Lines 1-3: Alphanumeric documentation 42 char/line.

Line 4: Cruise no., station no., absolute no. of 1st segment of the output file, input tape no., year of cast (all integer).

Line 5: Time interval between samples in seconds, zero, starting latitude in decimal degrees, starting longitude, time of fix a start of station in decimal days, ending latitude, ending longitude, ending time, maximum pressure of profile in decibars. (all floating point)

Line 6: Magnetic variation, magnetic dip, ship speed, ship heading (floating point).

Line 7: Dry bulb temperature, wet bulb temperature, surface temperature, wind speed in knots, wind direction in degrees, significant wave height in feet (floating point).

Line 8: Cast start day, hour, minute, second zero, cast end day, hour, minute, second (integer).

Line 9: Sequential file no. (integer)

Line 10: Station Identification No. consisting of a 3 digit station and a 3 digit sequential cast No.

Line 11: Message level 0-9

OUTPUT: File 20 contains the output FEB file.

**ADDITIONAL
INFORMATION:**

PROGRAM: CONVERSION

PURPOSE: To convert from instrument orthogonal to geomagnetic orthogonal coordinates.

MACHINE: UNIVAC 1108

LANGUAGE: FORTRAN V

AUTHOR: James J. Vega, corrected and modified by Kim David Saunders

FILE LOCATIONS: Absolute, Relative, and Symbolic
Elements - VEGA*LIB.CONVERSION

INPUT: File 10 - input FEB File.

OUTPUT: File 20 - output FEB File.

**ADDITIONAL
INFORMATION:**

PROGRAM: PREFIX

PURPOSE: To read a FEB file containing raw profiler data, removing points where the vertical velocity has exceeded the limits for the instrument.

MACHINE: UNIVAC 1108

LANGUAGE: FORTRAN V

AUTHOR: Kim David Saunders (April 1982)

FILE LOCATIONS: Absolute, Relocatable, and Symbolic Elements - VEGA*LIB.PREFIX

INPUT: Line 1) NUIN1,NSEG1,NSSEG1
 Line 2) NOUT
 NUIN1 = unit number of input FEB file
 NSEG1 = number of segments to be read
 NSSEG1 = number of first segment
 NUOUT = unit number of output FEB file

OUTPUT: The output FEB file has the same structure as the input file, with the exception that variable 14 now contains a rough approximation to the vertical instrument velocity estimated from the time derivative of the pressure. The initial time of the cast (in Julian days has been replaced).

ADDITIONAL INFORMATION: The NBIS 3VCTD profiler measures the current relative to the instrument by means of a three axis acoustic velocimeter. The operation of this type of current meter is described in the NBIS Acoustic Current Meter manual. The point of interest, here, is that the apparent velocity along any axis is proportional to the phase difference of the two acoustic pulses, which are, in turn, proportional to the true component of the water velocity along the axis. Thus, in principle, the measured velocity should be mapped onto the interval from about -100 to 100 cm/sec (nominal). In practice, this does not occur exactly, as when the phase of the acoustic signal is near -180 or 180 degrees, the gate opening/closing signals become ambiguous. This results in random output velocities when the true velocity component along the axis is within a small "dead band" of the velocity extremes. This program is designed to search the data for probable occurrences of this over-ranging and to correct (as much as possible) by substituting the projection of the vertical instrument velocity, determined by the time derivative of the pressure, for the components along the q1 and q3 axis.

PROGRAM: VF1X1

PURPOSE: Corrects the vertical component of current velocity as measured by the NORDA 3-component profiler.

INPUT: (Free format)

Line 1 - Input file specifications
IUNR - Unit no. for input
NSEG - No. of segments to process
NSSEG - No. of starting segment
MSGR - Message level for input
Line 2 - Output file specifications
IUNW - Unit no. for output
MSGW - Message level for output

NOTES:

- 1) The input file is presumed to be in geomagnetic coordinates as produced by program CONVERSION.
- 2) Input variables are identified by name as follows:

Vertical current speed, uncorrected - VLOG3
Vertical component of acceleration - ACCLG3
Pressure - PRESS
Time - RELSEC
- 3) The output file has the same structure as the input file except that the corrected vertical velocity, named W, is inserted in each data cycle immediately before the uncorrected vertical velocity, and the instrument velocity, named WI, is inserted in each data cycle immediately after the last variable (RELSEC).

METHOD: An estimate of vertical instrument velocity w_i is found from the observed acceleration and pressure. Derivation of w_i is given in the Appendix. The corrected velocity w is then obtained from the observed velocity w_0 by

$$w = w_0 - w_i$$

PRINTED OUTPUT: For each output segment, the following quantities are listed:

- o Start and end times (RELSEC)
- o Start and end pressure
- o $\alpha, \beta, \gamma, \lambda$ (see Appendix for definitions)

MACHINE: UNIVAC 1108

FILE LOCATIONS: Absolute element CODE331*FCHFILE1.VFIX1
Mapping element CODE331*FCHFILE1.MVFIX1

PROGRAM: TSERPLOT2

PURPOSE: To plot time series of profiler variables either versus cycle number or relative time.

MACHINE: UNIVAC 1108

LANGUAGE: FORTRAN V

AUTHOR: James J. Vega

FILE LOCATIONS: Absolute, Relocatable, and Symbolic

INPUT: There are two input lines for each subplot:

Line 1: IU, IABSIS, IP, YMAX, YSTP, YMIN, IDEC

Line 2: if IABSIS=0: NUMSEG, IBEGIN, CYCIN
if IABSIS=1: TSTART, TEND, IPTIME

where

IU = logical unit for input FEB file
IABSIS= 0 for cycle number plot
= 1 for time plot
IP = position of variable in FEB array
YMAX = expected max. of variable
YSTP = labling interval
YMIN = expected min. of variable
IDEC = decimation ratio

NUMSEG= number of segments to plot
IBEGIN= first segment to plot
CYCIN = number of cycles per inch

TSTART= start time in seconds (relative)
TEND = stop time in seconds
IPTIME= location in FEB array of time

OUTPUT: File 25 - output intermediate plot file.

ADDITIONAL
INFORMATION:

1. The following data should succeed the last data line to ensure proper termination of the program:

99,0,0,0.,0.,0.,0 .

2. This program is designed to use 34 inch ZETA plotting paper. Because of this, the maximum number of variables per plot is 5.

PROGRAM: VF1X1-S

PURPOSE: To read a FEB file containing profiler data and create a new FEB file which contains a corrected vertical velocity computed from the pressure derivative.

MACHINE: UNIVAC 1108

AUTHOR: Fred Hamrick (Apr11 82)

FILE LOCATIONS: Absolute element CODE331*FCHFILE1.VF1X1-S
Mapping element CODE331*FCHFILE1.MVF1X1-S

INPUT:

Line 1) IUNR,IUNW,NSEG,NSSEG,NVAR,NUMV
 IUNR = unit number for input FEB file
 IUNW = unit number for output FEB file
 NSEG = number of segments to read
 NSSEG= start segment
 NVAR = variable number for pressure
 NUMV = variable number for vertical velocity

OUTPUT:

The output FEB file has the same structure as the input file with two additional variables. The variable WI (instrument velocity) is written as the last variable, and the corrected velocity $W=V-WI$ (where V =measured vertical velocity) is written as the variable immediately before the measured vertical velocity.

ADDITIONAL INFORMATION:

- 1) Before executing this program, the input FEB file should be interpolated with respect to time. (This time difference is set in the program as variable DELT.) The interpolation may be performed by executing HTP*PROG.ZINTERP.
- 2) The instrument velocity is computed as

$WI = RHOG1 * DPDT$

Where $RHOG1 = .9955$

and $DPDT =$ pressure derivative
 is computed as:

$$DPDT(J) = \frac{3}{K(K+1)(2K+1)*DELT} \sum_{i=-K}^{i=+K} (i * P_{j+1})$$

Where P_j = pressure values

and K is set to 8 in the program.

(the first and last K values of $DPDT$ are set to 0).

PROGRAM:

UNORTHOG

PURPOSE:

To convert from instrument orthogonal coordinates to instrument acoustic axes coordinates. (Velocity only).

MACHINE:

UNIVAC 1108

LANGUAGE:

FORTRAN V

AUTHOR:

Kim David Saunders

FILE LOCATIONS:

Absolute, Relocatable and Symbolic
 Elements- VEGA*LIB.UNORTHOG

INPUT:

From terminal (unit 5)

Line 1: NUIN, NSEG, NSSEG

Line 2: NUOUT

NUIN = Input unit No.

NSEG = No. of Segments desired

NSSEG= No. of first segment.

NUOUT= Output unit No.

APPENDIX C: COMPUTER LISTINGS

```

0: "FILE 4 - TAPE ID V002";
1: "Profile Data logger-Part 2".ldt 2
2: "LATEST MOD: 02 DEC 1981.1645Z(KDS)";
3: "Added second depth profile subroutine";
4: " Added pressure Corection";
5: "Changed profiler from unit 8 to unit 5 - 8 now res. for disk";
6:
7: "0 array controls output options";
8: " [1] contains Hazeltine list option";
9: " [2] contains 7245A time series plot option";
10: " [3] contains 7245A depth plot option";
11: " [4] lists summaries of instrument motion statistics";
12: " [5] produces a second depth profile on unit 703";
13: " [6] produces a plot of one variable vs the other";
14: "Options 3 and 4 disable each other to prevent device conflict";
15:
16: "Function keys";
17: " F0- solicit display options";
18: " F0 start/stop data logging";
19: " F1 Generate end of data file";
20:
21: "Flags";
22: " 0 - if set, log to mag tape; if clear, don't";
23: " 1 - if set, close the data file; if clear, continue logging";
24: " 5 - if the matrices in the DISPLAY routines have been";
25: " - dimensioned, flg 5 is set. ";
26: " 6 - if set, the first pass through the TIMESERIESPLOT routine";
27: " 7 - first pass flag for INSTVEL ";
28: " 8 - if set, the 2 nd pass through the TIMESERIESPLOT routine";
29:
30: dim A(66,3),O(9),Z(38),T(27),R(0:10),S(1:);
31: dim V(66,20),V(10,0:10),Q(3),U(2),T(60)
32: dim B(3,3),F(4,3),O(6),E(6)
33: "LOAD VELOCITY ORTHOGONALIZATION MATRIX";
34: -.055+*F(1,1),-.803+*F(2,1),1.58+*F(3,1),-.794+*F(4,1)
35: -.197+*F(1,2),-.634+*F(2,2),-.011+*F(3,2),.759+*F(4,2)
36: .612+*F(1,3),.349+*F(2,3),-.071+*F(3,3),.309+*F(4,3)
37: trk 0,ldf 5,V(1),ldf 6,V(1),trk 0
38: 0-A;ent "Enter the Cast No.",A(1,1)
39: ent "Enter the Latitude in decimal degrees",A(6,1)
40: ent "Enter Longitude-decimal degrees",A(7,1)
41: cos(52)+A(11,1),-sin(52)+A(12,1)
42:
43: -3+O(1),0+O(3),0+O(2)
44: on err "ERRORHANDLER"
45: dev "nb",520
46: buf "b2",38,1
47: buf "b1",38,1,time 15000,fxd 0
48: "START";if flg1,ldp 7
49: buf "b1"
50: cmd 5," "
51: tfr "nb","b1",38
52: jmp rds("b1")-38
53: wait 200
54: buf "b2"
55: cmd 5," "
56: tfr "nb","b2",38
57: jmp rds("b2")-38
58: if flg0,gto "next"
59: cmd 5," "
60: cmd 5,"?"
61: "next",for i=1 to 38,rdb("b1")+Z(1),next i
62: if Z(1)-79,cli 8,prt "ERR",gto "START"
63:
64: cli "BREAKOUT"

```

```

65: for i=1 to 66,A[i,1]+A[i,2],next i
66: for i=1 to 38,rdp("b2")+Z[i],next i
67: if Z[i]-79,cli 8,prt "ERP",goto "START"
68: cli "BREAKOUT"
69: cli "OPTIONS"
70: cli "DERIVE"
71: cli "DISPLAY"
72:
73: if abs(A[15,1]-A[15,3])<1,goto "START"
74: for i=1 to 66,A[i,1]+A[i,3],next i
75: goto "START"
76:
77:
78: "BREAKOUT":
79: "Decode byte string Z into proper slots in A":
80: "PRESS ",(256Z[3]+Z[2])/200-A[15,1],if bit(0,Z[10]),-A[15,1]-A[15,1]
81: "PRESS CORR",.9989789716A[15,1]+.30325283-A[15,1]
82: "TEMP-S", (256Z[5]+Z[4])/2000+A[16,1]
83: "COND", (256Z[7]+Z[6])/1000+A[18,1]
84: "F-TEMP", (256Z[9]-Z[8])/2000+A[17,1]
85: "SPECIAL FIX FOR GYRE CPUISE ONLY",A[17,1]+A[16,1]
86: for i=1 to 27 by 2,shf(Z[i+1],-6)+shf(Z[i],2)+T[i]
87: if bit(0,Z[i])=1,-T[i]-T[i]
    next i
    "VELOC ":for i=1 to 3,T[11+2(i-1)].00639-A[22-i,1],next i
90: -A[20,1]+A[20,1]
91: "MAGNET":for i=1 to 3,T[17+2(i-1)]/1000+A[21+i,1],next i
92: -A[22,1]+A[22,1]
93: "ACCEL ":for i=1 to 3,T[23+2(i-1)]/1000+A[24+i,1],next i
94: -A[26,1]+A[26,1]
95: "TIME":
96: shf(Z[38],4)+T[1],Z[38]-shf(T[1],-4)+T[2]
97: shf(Z[37],4)+T[3],Z[37]-shf(T[3],-4)+T[4]
98: shf(Z[36],4)+T[5],Z[36]-shf(T[5],-4)+T[6]
99: shf(Z[35],4)+T[7],Z[35]-shf(T[7],-4)+T[8]
100: shf(Z[34],4)+T[9],Z[34]-shf(T[9],-4)+T[10]
101: shf(Z[33],4)+T[11],Z[33]-shf(T[11],-4)+T[12]
102: 100T[1]+10T[2]-T[3]+T[13]
103: 10T[4]+T[5]+T[14]
104: 10T[6]+T[7]-T[15]
105: 10T[8]+T[9]+T[10]/10+T[11]/100+T[12]/1000+T[16]
106: for i=1 to 4,T[i+12]+A[i+1,1],next i
107: ret
108:
109: "DERIVE":
110:
111: "SALINITY"(A[15,1],A[16,1],A[18,1])+A[28,1]
112: cli "SIGMA-T"(A[16,1],A[28,1],A[30,1])
113: cli "SND SPEED"
114: cli "ANGLE"
115: cli "TRANSFORMS"
116: cli "TRANSFORM Q"
117: cli "GMAG CURRENT"
118: cli "GMAG ACCEL"
119: cli "GMAG MAG"
120: cli "INST VEL"
121: cli "BEST VEL"
122: if abs(A[15,1]-A[15,3])<1,ret
123: cli "dp/dz"
124: cli "VEL SHEARS"
125: cli "N AND RI"
126: ret
127:
128: "SALINITY":
129: "ARGUMENTS: 1-A[15,1],2-A[16,1],3-A[18,1]":
130: 1.45038/9.9e7+p4
131: 6.76583621732e5+p5,2.00529363371e2+p6
132: 1.11098951812e-2+p7,-7.26681983148e-7+p8
133: 1.3586827285e-11+p9
134: p3*(1-5.25e-6*(p2-15)+p1p4)/42.906+p11
135: 1.60836e-5p1-5.4845e-10p1+2+6.166e-15p1+3+p12

```

```

136: p12/(1+.030786p2+3.169e-4p2+2)+p13
137: p11/(1+p13)+p14
138: 100p2+p10,(p5+p6p10+p7p10+2+p8p10+3+p9p10+4)(1+int(-6))+p15;p14/p15+p16
139: -.08996+28.8567p16+12.18882p16+2-.61869p16+3+p17
140: p17+5.98624p16+4-1.3231p16+5+p1
141: p17+p16(p16-1)(.0442p2-.07046p2+2-.004p16p2)+p18
142: ret p18
143:
144:
145: *SIGMA-T*,
146: *ARGUMENTS,1-A[16,1],2-A[29,1]*,
147: *VARIABLES DESTROYED : C[1],I,J*,
148: if not flg10,dim C[0,4,0,4];sfq 10
149: for I=0 to 3
150: for J=0 to 3
151: 0+C[I,J]
152: next J
153: next I
154: 8.00969062e-2+C[0,0];7.97018644e-1+C[0,1]
155: 1.31710842e-4+C[0,2];-6.11831499e-8+C[0,3]
156: 5.8819403e-2+C[1,0];-3.25310441e-3+C[1,1];2.8797153e-6+C[1,2]
157: -8.1465413e-3+C[2,0];3.89187483e-5+C[2,1]
158: 4.76600414e-5+C[3,0]
159: 0+p3
160: for I=0 to 3
161: for J=0 to 3
162: if I+J<4,p3+C[I,J]*0.1+I*p2+J+p3
163: next J
164: next I
165: ret
166:
167: *SND SPEED*,
168: *SOUND SPEED FORMULA*,
169: A[16,1]*X,A[29,1]*S,A[15,1]*D
170: 100(1449+4.6X-.055XX+.0003X+3+(1.39-.012X)(S-35)+.017D)+C
171: C+A[29,1]
172: re:
173:
174: *TRANSFORMS*,A[49,1]+p1
175: deg,cos(p1)+p2,-sin(p1)+p3
176: 0+X+Y
177: for I=1 to 3,A[I+21,1]+2+X+X,A[I+24,1]+2+Y+Y
178: next I
179: JX+X,JY+Y
180: for I=1 to 3,A[I+21,1]/X+E[1],A[I+24,1]/Y+E[1+3]
181: next I
182: for I=1 to 3,-E[1+3]+B[3,1],next I
183: for I=1 to 3,(E[1]-p3B[3,1])/p2+B[2,1],next I
184: sgn(A[22,1])*(1-B[2,1]+2-B[3,1]+2)+B[1,1]
185: A[26,1]A[24,1]-A[27,1]A[23,1]+p1
186: -B[1,1](A[25,1]A[24,1]-A[27,1]A[22,1])/p1+B[1,2]
187: B[1,1](A[25,1]A[23,1]-A[26,1]A[22,1])/p1+B[1,3]
188: ret
189:
190: *TRANSFORM Q*,
191: for J=1 to 3,F[1,J]*X
192: for I=1 to 3,A[I+18,1]F[I+1,J]*X+X;next I
193: X+A[J+30,1];next J
194: ret
195:
196: *GMAG CURRENT*,
197: *CURRENTS IN GEOMAG. COORD. FROM INSTRUMENT COORD.*.
198: for I=1 to 3
199: 0+Y
200: for J=1 to 3,B[I,J]A[J+30,1]+Y+Y;next J
201: Y+A[I+39,1]
202: next I
203: ret
204:
205: *GMAG ACCEL*,
206: *ACCELERATIONS, LESS G, IN GEOMAGNETIC COORDINATES*,

```

```

207: for I=1 to 3;0+X
208: for J=1 to 3;B[I,J]A[J+24,1]+X+X;next J
209: X+A[I+33,1];next I
210: A[36,1]+9.9+A[36,1]
211: ret
212:
213: "GMAG MAG";
214: "MAGNETIC FIELD IN GEOMAGNETIC COORDINATES";
215: for I=1 to 3;0+Y
216: for J=1 to 3;B[I,J]A[J+21,1]+Y+Y;next J
217: Y+A[I+36,1];next I
218: ret
219:
220: "ANGLE";
221: "COMPUTATION OF ANGLE BET. G' AND H'";
222: J(A[22,1]+2+A[23,1]+2+A[24,1]+2)+H+A[50,1]
223: A[22,1]/H-D[1];A[23,1]/H-D[2];A[24,1]/H-D[3]
224: J(A[25,1]+2+A[26,1]+2+A[27,1]+2)+G+A[51,1]
225: A[25,1]/G-D[4];A[26,1]/G-D[5];A[27,1]/G-D[6]
226: 0+R;for J=1 to 3;D[J]D[J+3]+R+R;next J
227: asn(B)+B;B+A[49,1]+A[9,1]
228: ret
229:
230: "INST VEL";
231: "INSTRUMENT VELOCITY";
232: A[2,1]-p1
233: (A[2,1]-p1)86400+3600A[3,1]+60A[4,1]+A[5,1]+p3
234: (A[2,2]-p1)86400+3600A[3,2]+60A[4,2]+A[5,2]+p2
235: if flg7;p3-p2+A[55,1]
236: if A[55,1]-0.5+A[55,1]
237: if not flg7;5+A[55,1];sfg 7
238: 100(A[15,1]-A[15,2])/A[55,1]+A[45,1]
239: 0+A[43,1]+A[44,1]
240: ret
241:
242: "BEST VEL";
243: "CORRECTS OBSERVED VELOCITIES FOR INSTRUMENT MOTION";
244: for I=1 to 3;A[I+39,1]-A[I+42,1]-A[I+45,1];next I
245: ret
246:
247: "dp/dz";
248: "VERTICAL DENSITY GRADIENT (dp/dz)";
249: A[15,1]-A[15,3]+A[56,1]
250: (A[30,1]-A[30,3])/A[56,1]+A[57,1]
251: ret
252:
253: "VEL SHEARS";
254: "VERTICAL VELOCITY SHEARS ";
255: for I=46 to 48;(A[I,1]-A[I,3])/A[56,1]+A[I+12,1]
256: ret
257:
258: "N AND R1";
259: "N AND RICHARDSON NUMBER";
260: 980A[57,1]/(1+A[30,1]/1000)+p1;if p1<0;-p1+p1
261: fp1+A[62,1]
262: A[58,1]+2+A[59,1]+2+p2
263: if p2#0;p1/p2+A[61,1]
264: if p2=0;999+A[61,1]
265: ret
266:
267: "DISPLAY";
268: cll "HAZELTIME"
269: cll "TIMESERIES"
270: cll "DEPTH"
271: cll "DEPTH2"
272: cll "V1V2"
273: cll "STATS"
274: ret
275:
276: "HAZELTIME";if O[1]<0;abs(O[1])+O[1];ccl "HAZINITIALIZE"
277: if O[1]=0;ret

```

```

278: cli 'HAZWRITE'
279: ret
280: 'HAZINITIALIZE':
281: if not flg5,dim L8[3,80],sfg 5
282: '=-L8[1]
283: for I=1 to V[0[1],0],V8[V[0[1],1],1,6]+L8[1,1+6(I-1),6]],next I
284: wtb 3,27,26;wtb 3,27,17,0,3
285: fmt 1,c80,z;wrt 3,L8[1];ret
286: 'HAZWRITE':
287: fmt 1,f6.2,z;fmt 2,/
288: wtb 3,27,17,0,5;wtb 3,27,26
289: for I=1 to V[0[1],0];wrt 3.1,A[V[0[1],1],1],next I
290: wrt 3.2;wtb 3,27,17,40,2
291: fmt 3,f3.0,2x,f2.0,2x,f2.0,2x,f6.3
292: wrt 3.3,A[2,1],A[3,1],A[4,1],A[5,1]
293: wtb 3,27,17,2,2
294: fmt 4,"CAST # ",f4.0;wrt 3.4,A[1,1]
295: wtb 3,27,17,2,1;fmt 5,"LATITUDE ",f6.3," LONGITUDE ",f8.3
296: wrt 3.5,A[6,1],A[7,1]
297: ret
298:
299: 'DEPTH':if 0[3]<0;-0[3]+0[3],0+0[4];cli 'DEPTHINITIALIZE'
300: if 0[3]=0;ret
301: cli 'DEPTHPLOT'
302: ret
303: 'DEPTHINITIALIZE':wtb 706,27,85
304: wrt 705,"IP,1000,1000,6000,6000";psc 705;pcir
305: fxd 0
306: csiz 3.5;sc1 0,10,0,10;plt 0,16,1;lbl T$
307: csiz 2;sc1 0,1,300,0;xax 0;xax 300;yax 1,20;yax 0,20,0,300,5
308: csiz 3,2,1,30;plt -.1,170,1;lbl 'Pressure';csiz 3,2,1,0
309: for I=1 to V[0[3],0]
310: val(V8[V[0[3],1],7,13])+p1
311: val(V8[V[0[3],1],14,20])+p2
312: A[15,1]+R[0],A[V[0[3],1],1]+R[1]
313: 5+p6;if abs(p2-p1)<10;1+p6
314: 1+p7;if abs(p2-p1)>99;50+p7;1+p6
315: csiz 2;sc1 p1,p2,0,10;xax 10+1,p7,p1,p2,p6
316: csiz 3;sc1 0,80,0,10;plt 30,10.1+1,1,1;lbl V8[V[0[3],1],1,6]
317: csiz 1.5
318: next I
319: ret
320: 'DEPTHPLOT':psc 705
321: for I=1 to V[0[3],0]
322: val(V8[V[0[3],1],7,13])+p1
323: val(V8[V[0[3],1],14,20])+p2
324: sc1 p1,p2,300,0;lim p1,p2,300,0
325: A[V[0[3],1],1]+p3;A[15,1]+p4
326: plt p3,p4,1;plt p3,p4,2
327: p3+R[1],p4+R[0]
328: next I
329: lim
330: ret
331:
332: 'STATS':if 0[4]=0;ret
333: if 0[4]<0;-0[4]+0[4],0+0[3];cli 'STATSINITIALIZE'
334: cli 'STATSREQ'
335: ret
336: 'STATSINITIALIZE':wtb 706,27,85
337: wrt 706,T$
338: for I=1 to 7;0-S[1];next I
339: fmt 1,20x,"INSTRUMENT ATTITUDE STATISTICS",/
340: fmt 2,5x,"PITCH",14x,"YAW",13x,"MAG DIP",13x,"PRESSURE"
341: fmt 3," Mean Std Dev Mean Std Dev Mean Std Dev Mean",/
342: wrt 706.1;wrt 706.2;wrt 706.3
343: ret
344: 'STATSREQ':S[1]+1+S[1]
345: 'Pitch':deg;acs(B[3,3])+X;cli 'SUPDATE1'(2,X)
346: 'Yaw':acs(B[1,1])+X;cli 'SUPDATE1'(4,X)
347: 'MagDip':A[49,1]+X;cli 'SUPDATE1'(6,X)
348: 'Press':A[19,1]+X;cli 'SUPDATE1'(8,X)

```

```

349: if S[1]-20, goto "SLIST"
350: ret
351: "SLIST": for i=2 to 8 by 2, goto "SUPDATE2"
352: next i
353: fmt 1, f7.2, 2x, z; fmt 2, /
354: for i=2 to 8; wrt 706.1, S[i]; next i; wrt 708
355: for i=1 to 7; 0+S[i]; next i
356: ret
357: "SUPDATE1": S[p1]+p2+S[p1], S[p1+1]+p2+2+S[p1+1]; ret
358: "SUPDATE2": S[i]/S[i]+S[i], sqrt(S[i+1]/S[i]-S[i]+2)+S[i+1]; ret
359:
360: "TIMESERIES": if D[2]<0, -0[2]+0[2], 0+0[5]; call "TIMESERIESINITIALIZE"
361: if 0[2]=0; ret
362: call "TIMESERIESPLOT"
363: ret
364: "TIMESERIESINITIALIZE": wtb 704, 27, 85
365: fxd 0; cfg 8
366: ps: 703; pclr
367: wrt 703, "IP, 1000, 1000, 7500, 10000"
368: csiz 2
369: scl 0, 2000, 0, 1; xax 0, 100, 0, 2000, 5; yax 2000; xax 1; yax 0
370: V[0[2], 0]+p1
371: for i=1 to p1-1; xax i/p1, 100; next i
372: csiz 2, 2, 1, 90
373: for i=1 to p1; plt -200, (i-1)/p1+.02, 1; lbl V[V[0[2], i], 1, 6]; next i
374: 9000/p1+p2
375: for i=1 to p1; fmt 1, "IP", " ", fz4.0, " ", fz4.0, " ", fz5.0, " ", fz5.0
376: wrt 703.1, 1000, 1000+(i-1)p2, 7500, 1000+ip2
377: csiz p1, 2, 1/6.5, 0
378: val(V[V[0[2], i], 7, 13])+p3
379: val(V[V[0[2], i], 14, 20])+p4
380: scl 0, 2000, p3, p4; fxd 0, 5+p6; if abs(p4-p3)<10; 1+p6
381: 1+p7; if abs(p4-p3)>50; 2+p6; 50+p7
382: if lmod2; yax 0, p7, p3, p4, p6
383: if not lmod2; yax -80, p7, p3, p4, p6
384: next i
385: ret
386: "TIMESERIESPLOT": psc 703
387: if not flg6 or flg8; goto "TSnext"
388: A[3, 1]+Q[3], A[4, 1]+Q[4], A[5, 1]+Q[5]
389: A[2, 1]+Q[1], 3600A[3, 1]+60A[4, 1]+A[5, 1]+Q[2]; sfg 8
390: wrt 703, "IP, 1000, 1000, 7500, 10000"; scl 0, 80, 0, 10
391: fxd 0; csiz 1.5
392: plt 10, 10.3, 1; lbl "Start Time ", Q[3], " ", Q[4], " ", Q[5]
393: plt 10, 10.5, 1; lbl "Start Day ", Q[1]
394: "TSnext":
395: if flg6; goto "TSnext"
396: dim Q[7]; A[2, 1]+Q[1], 3600A[3, 1]+60A[4, 1]+A[5, 1]+Q[2]
397: sfg 6
398: "TSnext":
399: 0+p8; if A[2, 1]>Q[1], 86+00+p8
400: V[0[2], 0]+p1; 9000/p1+p2
401: p8+3600A[3, 1]+60A[4, 1]+A[5, 1]-Q[2]+p9
402: for i=1 to p1; fmt 1, "IP", " ", fz4.0, " ", fz4.0, " ", fz5.0, " ", fz5.0
403: wrt 703.1, 1000, 1000+(i-1)p2, 7500, 1000+ip2
404: val(V[V[0[2], i], 7, 13])+p3
405: val(V[V[0[2], i], 14, 20])+p4
406: scl 0, 2000, p3, p4; lim 0, 2000, p3, p4
407: A[V[0[2], i], 1]+p10
408: plt p9, p10, 1
409: plt p9, p10, 2
410: next i
411: lim
412: ret
413: "ERRORHANDLER": if ern=4; prt "TIMEOUT ERROR"
414: prt "ERN ", ern
415: prt "Line", erl
416: prt "ROM", rom
417: time 0
418: cli 7
419: cli 5

```

```

420: gto "START"
421:
422: "OPTIONS":if A=0;ret
423: ent "Enter Display Title",T8
424: "YESYES":ent "Enter Display Device No.",A;if A=0;ret
425: if A>9 or A<0;jmp -1
426: ent "Enter display option no.",O[A];if O[A]>0;-O[A]+O[A]
427: ent "Do you want more?",Q8;if cap(Q8[1,1])="Y";gte "YESYES"
428: if cap(Q8[1,1])="Y" and cap(Q8[1,1])="N";jmp -1
429: O=A;ret
430:
431: "SIGTEST":fmt 1,3f10.5
432: ent "enter T",T;ent "enter S",S
433: cll "SIGMA-T"(T,S,Q)
434: wrt 3.1,T,S,Q
435: gto "SIGTEST"
436: "TSTST":cll "TIMESERIESPLOT"
437: slp
438: "DEPTH2":if O[5]<0;-O[5]+O[5];0+O[2];cll "DEPTHINITIALIZE2"
439: if O[5]=0;ret
440: cll "DEPTHLOT2"
441: ret
442: "DEPTHINITIALIZE2":wtb 704,27,85
443: wrt 703,"IP,1000,1000,6000,6000";psc 703;pclr
444: fxd 0
445: csiz 3.5;sc1 0,10,0,10;plt 0,16,1;lbl T8
446: csiz 2;sc1 0,1,300,0;xax 0;xax 300;yax 1,20;yax 0,20,0,300,5
447: csiz 3,2,1,90;plt -.1,170,1;lbl "Pressure";csiz 3,2,1,0
448: for I=1 to V[O[5],0]
449: val(V8[V[O[5],1],7,13])>p1
450: val(V8[V[O[5],1],14,20])>p2
451: A[15,1]>R[0];A[V[O[5],1],1]>R[1]
452: 5>p6;if abs(p2-p1)<10;1>p6
453: 1>p7;if abs(p2-p1)>99;50>p7;1>p6
454: csiz 2;sc1 p1,p2,0,10;xax 10+1,p7,p1,p2,p6
455: csiz 3;sc1 0,80,0,10;plt 30,10.1+i,1;lbl V8[V[O[5],1],1.6]
456: csiz 1.5
457: next I
458: ret
459: "DEPTHLOT2":psc 703
460: for I=1 to V[O[5],0]
461: val(V8[V[O[5],1],7,13])>p1
462: val(V8[V[O[5],1],14,20])>p2
463: sc1 p1,p2,300,0;lim p1,p2,300,0
464: A[V[O[5],1],1]>p3;A[15,1]>p4
465: plt p3,p4,1;plt p3,p4,2
466: p3>R[1];p4>R[0]
467: next I
468: lim
469: ret
470: "V1V2":if O[6]<0;-O[6]+O[6];cll "V1V2INIT"
471: if O[6]=0;ret
472: cll "V1V2PLOT"
473: ret
474: "V1V2INIT":ent "Enter the plot unit",Z
475: if Z=703;0+O[5]+O[2]
476: if Z=705;0+O[4]+O[3]
477: wtb Z+1,27,85
478: fxd 0;psc Z
479: pclr
480: wrt Z,"IP,1000,1000,6000,6000"
481: ent "Variable number 1",U[1]
482: ent "Variable number 2",U[2]
483: csiz 2
484: val(V8[U[1],7,13])>p1;val(V8[U[1],14,20])>p2
485: val(V8[U[2],7,13])>p3;val(V8[U[2],14,20])>p4
486: sc1 p1,p2,p3,p4;5>p6;if abs(p2-p1)<10;1>p6
487: 1>p7;if abs(p2-p1)>99;50>p7;1>p6
488: xax p3,p7,p1,p2,p6;xax p4
489: 5>p6;if abs(p4-p3)<10;1>p6
490: 1>p7;if abs(p4-p3)>99;50>p7;1>p6

```

```

491: yax p1,p7,p3,p4,p6,yax p2
492: csiz 3;scl 0,80,0,10;plt 30,11.1,1,1bl V$[U(1),1,6]
493: csiz 3.5;scl 0,10,0,10;plt 0,15,1,1bl T$
494: scl 0,10,0,80;csiz 3,2,1,90;plt -1,30,1,1bl V$[U(2),1,6]
495: csiz ;lim
496: ret
497: *V1V2PLOT*.psc 2
498: val(V$[U(1),7,13])~p1;val(V$[U(1),14,20])~p2
499: val(V$[U(2),7,13])~p3;val(V$[U(2),14,20])~p4
500: scl p1,p2,p3,p4~lim p1,p2,p3,p4
501: A[U(1),1]~p5;A[U(2),1]~p6
502: plt p5,p6,1;plt p5,p6,2;lim
503: ret
#9743

```

VARIABLES IN DISPLAY OPTIONS

OPTION # 1

1	DAY
2	HP
3	MIN
4	SEC
5	CAST
6	LAT
7	LONG

OPTION # 2

1	PRESS
2	U1-GC
3	U2-GC
4	U3-GC
5	U13-GC
6	ABSG-0
7	ABSH-0
8	MAGDIP

OPTION # 3

1	T-SLOW
2	PRESS
3	A1-GC
4	A2-GC
5	A3-GC
6	P1-GC
7	H2-GC
8	H3-GC

OPTION # 4

1	PRESS
2	U1-IC
3	U2-IC
4	U3-IC
5	Q1-AP
6	Q2-AP
7	Q3-AP
9	THETA
9	PHI

OPTION # 5

1	PRESS
2	U1-GC
3	U2-GC
4	U3-GC
5	ABSG-0
6	N
7	RI
8	T-SLOW
9	SAL

OPTION # 6

1	T-SLOW
2	SAL
3	SIGMAT

OPTION # 7

1	G1-IC
2	G2-IC
3	G3-IC
4	H1-IC
5	H2-IC
6	H3-IC

OPTION # 8

1	U1-GC
2	U2-GC

IA*LIB(1).TRANSCRIBE

```

1 C*****
2 C*****
3 C*** PROGRAM : TRANSCRIBE ***
4 C*** PURPOSE : TRANSLATES PROFILER DATA TO ENGINEERING UNITS ***
5 C*** LEAVING VELOCITY DATA IN NON-ORTHOGONAL ***
6 C*** INSTRUMENT COORDINATES. ***
7 C*** ***
8 C*** AUTHOR : J.J. VEGA , COMPUTER SCIENCES CORP. (PRIMARY) ***
9 C*** K.D. SAUNDERS , NORDA ( SECONDARY - REQUIRED IN ***
10 C*** ORDER TO CORRECT VEGA'S CODE AND COMPLETE ***
11 C*** DOCUMENTATION.) ***
12 C*** ***
13 C*** DATE (OF LATEST REVISION) : 15 MARCH 1982 ***
14 C*** ***
15 C*** INPUT : FILE TYPE OF DATA/COMMENTS ***
16 C*** 5 INPUT FROM CARDS/TERMINAL ***
17 C*** (IF USED FROM A TERMINAL, THE PROGRAM ***
18 C*** SUPPLIES SOLICITATION PROMPTS.) ***
19 C*** 10 DATA TAPE IN HP 9825A / NORDA FORMAT ***
20 C*** (THE TAPE SHOULD BE COPIED FROM THE ***
21 C*** 800 BPI, FAW TAPE TO A HIGHER DENSITY ***
22 C*** TAPE FOR TWO REASONS: FIRST, THE ***
23 C*** PROGRAM WILL RUN FASTER AND, SECOND, ***
24 C*** YOU WILL NOT GET ABNORMAL FRAME COUNTS ***
25 C*** WHICH WILL TERMINATE THE PROGRAM! THE ***
26 C*** PROPER METHOD FOR COPYING IS USE ***
27 C*** @COPY,MN INPUT,10 .) ***
28 C*** ***
29 C*** OUTPUT ***
30 C*** ***
31 C*** 20 FEB FILE CONTAINING PROFILER DATA. ***
32 C*** (THIS FILE MUST BE ASSIGNED PRIOR TO ***
33 C*** EXECUTING THE PROGRAM.) ***
34 C*** ***
35 C*****
36 C*****
37 C
38 C
39 C MAIN CALLING ROUTINE FOR TRANSCRIBE ROUTINE
40 C
41 C THIS PROGRAM READS DATA FROM MAG TAPE,
42 C DECODES IT, AND STORES IT IN FEB FILES.
43 C
44 C
45 C ARRAY LIST
46 C DBLK IS THE INPUT DATA ARRAY
47 C READ FROM TAPE.
48 C RD IS THE ARRAY CONTAINING
49 C THE BYTES OF THE DATA
50 C STRING TO BE DECODED.
51 C VAR IS THE ARRAY CONTAINING
52 C THE DECODED VARIABLES OF
53 C A DATA STRING.
54 C VM IS THE FEB FILE DATA ARRAY
55 C ADOCM IS THE FEB FILE ALPHA-
56 C NUMERIC HEADER ARRAY.
57 C FPOCM IS THE FEB FILE FLOATING
58 C POINT HEADER ARRAY.
59 C IDOCH IS THE FEB FILE INTEGER
60 C HEADER ARRAY.
61 C
62 C SEE NAVO TECH NOTE 'THE FAST AND

```

```

63 C EASY BINARY (FEB) FILE BY
64 C Z.R. HALLOCK FOR MORE ON FEB
65 C FILES.
66 C
67 C COMMONS FOR ZURIT
68 C
69 COMMON/WHDR/LW,NW,NBW,NMBW,NMFU,NFU,NIV,NAW,IPW(15)
70 COMMON/WDATA/VW(15,500)
71 COMMON/WDOCF/FDOCF(40)
72 COMMON/WDOC I/IDOCW(20)
73 COMMON/WDOCA/ADOCW(50)
74 C
75 COMMON/DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNNA,IRST,INST
76 C
77 C INITIALIZE CONTROL HEADER
78 C
79 DATA LW,NW,NFU,NIV,NAW/15,500,40,20,51
80 DATA (IPW(I),I=1,15)/'PRESS','STEMP','COND','FTEMP',
81 S 'VLOC11','VLOC12','VLOC13','MAG11','MAG12','MAG13','ACCL11',
82 S 'ACCL12','ACCL13','TIME','RELSEC'
83 DATA NNNW,NNIP,NNF,NNI,NNNA/500,15,40,20,50/
84 C
85 C READ ADCC ARRAY
86 WRITE(6,5000)
87 READ (5,10) (ADOCW(I),I=1,7)
88 READ (5,10) (ADOCW(I),I=13,19)
89 READ (5,10) (ADOCW(I),I=25,31)
90 C
91 C READ IDOC ARRAY
92 WRITE(6,5001)
93 READ (5,40) (IDOCW(1),IDOCW(3),IDOCW(6),IDOCW(10),IDOCW(14))
94 C
95 C READ FDOC ARRAY
96 WRITE(6,5002)
97 READ (5,40) (FDOCF(I),I=1,9)
98 WRITE(6,5003)
99 READ (5,40) (FDOCF(I),I=16,19)
100 WRITE(6,5004)
101 READ (5,40) (FDOCF(I),I=21,27)
102 WRITE(6,5005)
103 READ (5,40) (FDOCF(I),I=31,39)
104 C
105 C READ FILE NAME, SEGMENT NAME, MSG LEVEL
106 WRITE(6,5006)
107 READ (5,50) NMFU
108 WRITE(6,5007)
109 READ (5,50) NMBW
110 WRITE(6,5008)
111 READ (5,40) MSGW
112 C
113 C
114 10 FORMAT (7A6)
115 40 FORMAT (I)
116 50 FORMAT (A6)
117 C
118 ADOCW(27)=' INST.'
119 ADOCW(38)=' COORD'
120 ADOCW(39)=' '
121 C CALL SUPROUTINE TRANSC
122 CALL TRANSC
123 C
124 C
125 5000 FORMAT(' ENTER 3 LINES OF ALPHAMERIC DOCUMENTATION')

```

```

126 5001 FORMAT( ' ENTER AS INTEGERS: CRUISE NO., ST. ID. NO., '
127 1 ' ABSOLUTE NO. OF 1ST SEG., INPUT TAPE NO., YEAR OF CAST' )
128 5002 FORMAT( ' ENTER (FLOATING PT.) '
129 1 ' SAMPLE INTERVAL (SEC), ZERO, STARTING LATITUDE (DEC. DEG.) /
130 2 ' STARTING LONGITUDE, TIME OF FIX AT START (DEC. DAYS) /
131 3 ' ENDING LATITUDE, ENDING LONGITUDE, ENDING TIME /
132 4 ' MAXIMUM PRESSURE (DECIRANS) ' / )
133 5003 FORMAT( ' ENTER (F) : MAGNETIC VARIATION AND DIP, '
134 1 ' SHIP SPEED AND HEADING ' / )
135 5004 FORMAT( ' ENTER DRY BULB TEMP., WET BULB TEMP., SURFACE TEMP. '
136 1 ' BAROMETRIC PRESSURE, WIND SPEED (KTS), WIND DIRECTION, '
137 2 ' SIGNIFICANT WAVE HEIGHT (FT) ' / )
138 5005 FORMAT( ' ENTER (F) : CAST START DAY, HOUR, MIN, SEC /
139 1 ' ZERO, CAST END TIME: DAY, HOUR, MIN, SEC / )
140 5006 FORMAT( ' ENTER THE SEQUENTIAL FILE NUMBER (I) '
141 5007 ' ENTER NMEW WHICH CONSISTS OF A 3 DIGIT CAST NO., '
142 1 ' AND A 3 DIGIT PROFILE NO. ' / )
143 5008 FORMAT( ' ENTER THE MESSAGE LEVEL G-9 ' / )
144 END

```

BPRT,S V.TRANSC

SA*LIB(1).TRANSC

```

1 SUBROUTINE TRANSC
2 C
3 C THIS ROUTINE READS SINGLE DATA BLOCKS FROM
4 C MAG TAPE AND CALLS SUBROUTINES TO FIND AND
5 C DECODE EACH DATA STRING. AFTER 500 DATA
6 C CYCLES ARE DECODED, A FEB SEGMENT IS WRITTEN.
7 C
8 C COMMONS FOR ZWRIT
9 C
10 COMMON/UNDR/LN,NK,NBY,NMBU,NMFU,NFU,NIU,NAU,2PW(15)
11 COMMON/WDATA/VW(15,500)
12 COMMON/WDOCF/FDOCV(40)
13 COMMON/WDOC1/IDOCV(20)
14 COMMON/WDOCA/ADOCV(50)
15 C
16 COMMON/DIAGS/MSGR,MSGU,NNNR,NNNU,NNIP,NNF,NNI,VNA,IRST,ZVST
17 COMMON/EETA/BMAT(3,3),ORTHOG(4,3),ACCAL(3)
18 C INITIALIZE VELOCITY ORTHOGONALIZATION MATRIX
19 DATA ORTHOG(1,1),ORTHOG(2,1),ORTHOG(3,1),ORTHOG(4,1) /
20 0.55,-.403,1.58,-.794 /
21 DATA ORTHOG(1,2),ORTHOG(2,2),ORTHOG(3,2),ORTHOG(4,2) /
22 -.197,-.684,-.011,-.759 /
23 DATA ORTHOG(1,3),ORTHOG(2,3),ORTHOG(3,3),ORTHOG(4,3) /
24 -.612,.849,-.071,-.809 /
25 C
26 INTEGER BLKSI7
27 DATA BLKSI7/495/
28 DATA I2,J,K,MFLG,NFLG,ID/640/
29 DIMENSION VAR(17),RD(37)
30 INTEGER DBLK(500)
31 DOUBLE PRECISION OPTIME,RELDAY
32 REAL W(31),LENGTH
33 NFLG=1
34 IDOCV(1)=0
35 IDOCV(4)=0
36 NMBYTE=1
37 N2=0
38 IFIRST=1

```

```

39          NBAD=0
40          ISEG = 0
41          NRDTOT = 0
42          C
43          C *****
44          C
45          10      CONTINUE
46          C
47          IF (KFLG.EQ.0) GO TO 30
48          C
49          C *****
50          C READ NEXT INPUT BLOCK.
51          20      CONTINUE
52          DO 1000 I = 1 ,500
53          DBLK(I) = 0
54          1000    CONTINUE
55          KFLG = 0
56          CALL NTRAN (10,2,414,DBLK,L1)
57          IF (L1.EQ.-1) CALL NTRAN (10,22)
58          IF (L1.EQ.-2) GO TO 22
59          IF (L1.EQ.-3.OR.L1.EQ.-4) GO TO 1020
60          KFLG=0
61          NMBYTE=1
62          GO TO 30
63          C
64          C *****
65          C IF EOF READ, WRITE PARTIAL SEGMENT AND TERMINATE ROUTINE.
66          22      NFLG=1
67          FDOCN(1)=1
68          WRITE (6,23)
69          23      FORMAT (' EOF READ BEFORE SPECIFIED END OF CAST TIME',/,
70          $ ' ** PARTIAL SEGMENT WRITTEN **')
71          C
72          C *****
73          C CALL ROUTINE TO READ AND DECODE DATASTRING
74          30      CONTINUE
75          CALL DATSTR (KFLG,DBLK,BLKSI2,NMBYTE,RD,VAR)
76          C
77          IF (KFLG.EQ.1 .AND. NFLG .NE. 1) GO TO 20
78          C
79          C *****
80          C
81          C
82          C DECODED DATA TO FEB DATA ARRAY
83          C
84          C CHECK FOR BAD CYCLE
85          C
86          C
87          C*****
88          C          CHECK THAT THE PRESSURE IS IN RANGE
89          C*****
90          C
91          IF (VAR(11).GT.FDOCN(91).OR.
92          $ VAR(11).LT.-5) GO TO 999
93          C
94          C*****
95          C          CHECK THAT THE DATE IS IN RANGE
96          C*****
97          C
98          IF (VAR(14).GT.FDOCN(36).OR.
99          $ VAR(14).LT.FDOCN(31)) GO TO 999
100         C

```

```

101 C*****
102 C CHECK THAT THE TIME FIELDS ARE EACH IN RANGE *
103 C*****
104 C
105 IF (VAR(15).GE.24.000) GO TO 999
106 IF (VAR(16).GE.60.000) GO TO 999
107 IF (VAR(17).GE.60.000) GO TO 999
108 IF (IFIRST.NE.1) GO TO 75
109 C
110 C COMPUTE DPTIME FOR FIRST CYCLE OF FIRST SEGMENT ONLY.
111 DPTIME=VAR(14)+VAR(15)/24.+VAR(16)/1440.+VAR(17)/86400.
112 IFIRST=0
113 C
114 75 CONTINUE
115 C
116 C COMPUTE RELSEC FOR ALL CYCLES.
117 RELDAY=VAR(14)+VAR(15)/24.+VAR(16)/1440.+VAR(17)/86400.
118 RELSEC=86400.*(RELDAY-DPTIME)
119 N2 = N2 + 1
120 C ORTHOGONALIZE VELOCITY VECTOR
121 C
122 DO 9910 J=1,3
123 XO=ORTHOG(1,J)
124 DO 5 I=1,3
125 5 XO=VAR(I+4)*ORTHOG(I+1,J)+XO
126 9910 W(J)=XO
127 DO 60 I=1,3
128 60 VAR(I+4)=W(I)
129 LENGTH = SQRT(W(1)*W(1)+W(2)*W(2)+W(3)*W(3))
130 WRITE(8,1999) LENGTH
131 1999 FORMAT(' LENGTH OF ABSOLUTE VELOCITY PRIOR TO TRANSFORM
132 1 ,G20.5)
133 WRITE(8,2999) W(1),W(2),W(3)
134 2999 FORMAT(' V BEFORE ',G20.7)
135 DO 150 I=1,13
136 150 I3=I
137 150 VW(I,N2)=VAR(I)
138 VW(I4,N2)=DPTIME
139 VW(I5,N2)=RELSEC
140 C
141 IF (N2.LT.500.AND.NFLG.NE.1) GO TO 10
142 IF (N2.LT.500 .AND. NFLG.EQ.1 .AND. NFLG.EQ.1)GOTO30
143 C
144 C *****
145 C WRITE FEB SEGMENT
146 C
147 40 IB=0
148 IF (NFLG.EQ.1) NW=N2
149 N2=0
150 IDOCW(4)=IDOCW(4)+1
151 IDOCW(13)=NBAD
152 FDOCW(10)=DPTIME
153 FDOCW(13)=FDOCW(36)+FDOCW(37)/24.+FDOCW(38)/1440.
154 5 +FDOCW(39)/86400.
155 CALL ZWRIT (20,IF,IB)
156 ISEG = ISEG + 1
157 IF (NFLG.EQ.1) RETURN
158 NBAD=0
159 C
160 C CLEAR DATA ARRAY
161 C
162 DO 200 I=1,15
163 DO 200 J=1,500

```

```

164 200  VW(I,J)=0.00
165 C
166 999  CONTINUE
167 C  COUNT AND OMIT PAD CYCLES.
168      NBAD=NBAD+1
169      NBDTOT = NBDTOT + 1
170 C
171 C *****
172 C  WRITE INFORMATION CONCERNING THE BAD RECORD TO UNIT 9
173 C *****
174 C
175 C
176      WRITE(9,8000) N2,ISEG,NBDTOT,VAR(1),VAR(14),VAR(15),VAR(16)
177 8000  FORMAT(3I5,4I6,5)
178 C *****
179 C  TEST FOR END OF CAST
180 C
181      IF(INFLG.EQ.1.AND.KFLG.EQ.0) GOTO30
182      IF(INFLG.EQ.1.AND.KFLG.EQ.1) GOTO40
183      IF(INFLG.NE.1) GO TO 10
184 C  IF END OF CAST, RETURN TO CALLING ROUTINE
185 C  AND TERMINATE FEB FILE.
186 C
187 C *****
188      RETURN
189 1020  WRITE(6,1021) L
190 1021  FORMAT(' L=',I3,' IN DATABLOCK READ.')
191      END

```

APRT,S V.DATASRING

SA=LIB(1).DATASRING

```

1  SUBROUTINE DATSTR (KFLG,INBUF,IBUFSZ,NMBYTE,RO,VAR)
2  C
3  C  THIS ROUTINE SCANS THE ARRAY DBLK, BY BYTE, FOR
4  C  A VALID FRAME SYNC NUMBER. WHEN ONE IS FOUND,
5  C  EACH BYTE OF THE DATA STRING IS TRANSFERED TO
6  C  A SEPARATE ELEMENT OF ARRAY RO, AND SUBROUTINE
7  C  DECODE IS CALLED TO DECODE THE STRING.
8  C
9  DIMENSION IFS(2),RO(37),VAR(17)
10 INTEGER FS,FS2,FS3
11 C
12 C *****
13 15  CONTINUE
14 C
15 C *****
16 C  SCAN FOR F,S.
17      CALL MOVE (INBUF,NMBYTE,IFS(1),1,1)
18      NMBYTE=NMBYTE+1
19      IF (NMBYTE .LE. 534) GOTO30
20      NMBYTE = 1
21      KFLG = 1
22      RETURN
23 30  FS=FLO(0,8,IFS(1))
24      IF (FS.NE.15.AND.FS.NE.240) GO TO 15
25      NPLUS = NMBYTE + 28
26      CALL MOVE(INBUF,NPLUS,IFS(1),1,1)
27      FS2 = FLO(0,8,IFS(1))
28      NPLUS = NMBYTE+8
29      CALL MOVE(INBUF,NPLUS,IFS(1),1,1)
30      FS3 = 0

```

```

31      FS3 = FLD(0,5,IFS(1))
32      IF(IFS2,NE,-255) GOTO 15
33      IF(IFS3,NE,31) WRITE(16,8777) IFS(1),FS3
34      8777  FORMAT(5X,012,I10)
35      IF(IFS3,NE,31) GOTO 15
36      C *****
37      C
38      C *****
39      C  MOVE NEXT 37 BYTES INTO RD ARRAY
40      NBYTE = NMBYTE
41      DO 20 I=1,37
42      CALL MOVE(INSUF,NBYTE,RD(I),1,1)
43      NBYTE=NBYTE+1
44      20  CONTINUE
45      C
46      C *****
47      IF (NMBYTE.GE.534) KFLG=1
48      IF(NMBYTE.GE.534) NMBYTE = 1
49      C
50      CALL DECODE (RD,VAR)
51      C *****
52      RETURN
53      END

```

APRT,S V.DECODE

JA*LIB(1),DECODE

1 SUBROUTINE DECODE (RD,VAR)

```

2  C
3  C THIS PROGRAM DECODES THE DATA BROKEN OUT
4  C BY SUBROUTINE DATSTR. THE DECODED VARIABLES
5  C ARE STORED IN ARRAY VAR.
6  C

```

```

7      PEAL RD(1),VAR(1)
8      INTEGER I,T(17)
9      C

```

10 C CONSTRUCT SCALAR WORDS

```

11 C
12      DO 5 I=2,8,2
13      T(I)=FLD(0,8,RD(I-1))
14      5  FLD(20,8,T(I))=FLD(0,8,RD(I))
15      C

```

16 C COMPUTE SCALAR QUANTITIES

```

17 C
18 C PRESSURE
19 C

```

```

20 C*****
21 C*****
22 C**
23 C** PRESSURE CALIBRATION OF 10/8/81 , 10/9/81 APPLIED **
24 C**
25 C*****
26 C*****
27 C

```

```

28      VAR(1)=T(2)/200.
29      VAR(1)= 0.9989789*VAR(1)+0.20325
30      IF (FLD(17,1,RD(9)),EQ,1) VAR(1)=-VAR(1)
31      C

```

32 C SLOW TEMP.

```

33      VAR(2)=T(4)/2000.
34      IF (FLD(16,1,RD(9)),EQ,1) VAR(2)=-VAR(2)

```

```

35      C
36      C CONDUCTIVITY
37      VAR(3)=T(6)/1000.
38      C
39      C FAST TEMP.
40      VAR(4)=T(8)/2000.
41      IF (FLD(5,1,RD(9)).EQ.1) VAR(4)=-VAR(4)
42      C
43      C CONSTRUCT VECTOR COMPONENT WORDS
44      C
45      DO 10 I=10,26,2
46      T(I-9)=FLD(0,6,RD(I))
47      FLD(22,6,T(I-9))=FLD(0,8,RD(I+1))
48      10 IF (FLD(7,1,RD(1)).EQ.1) T(I-9)=-T(I-9)
49      C
50      C COMPUTE VECTOR COMPONENTS
51      C
52      C VELOCITY VECTOR
53      C
54      C*****
55      C*****
56      C***
57      C*** NOTE : NO CALIBRATIONS HAVE BEEN APPLIED TO THE VELOCITY ***
58      C*** COMPONENTS IN INSTRUMENT COORDINATES. ***
59      C***
60      C*****
61      C*****
62      C
63      VAR(5)=T(11)*.0061039
64      VAR(6)=-T(13)*.0061039
65      VAR(7)=T(15)*.0061039
66      C
67      C MAGNETIC VECTOR
68      VAR(8)=-T(17)/1000.
69      VAR(9)=T(19)/1000.
70      VAR(10)=T(11)/1000.
71      C
72      C ACCELERATION VECTOR
73      VAR(11)=T(13)/1000.
74      VAR(12)=-T(15)/1000.
75      VAR(13)=T(17)/1000.
76      C
77      C COMPUTE CAST TIME
78      C
79      C JULIAN DAY
80      VAR(14)=100*FLD(0,4,RD(37))+10*FLD(4,4,RD(37))
81      S *FLD(0,4,RD(36))
82      C
83      C HOUR
84      VAR(15)=10*FLD(4,4,RD(36))+FLD(0,4,RD(35))
85      C
86      C MINUTE
87      VAR(16)=10*FLD(4,4,RD(35))+FLD(0,4,RD(34))
88      C
89      C SECOND
90      VAR(17)=10.*FLD(4,4,RD(34))+FLD(0,4,RD(33))
91      S *FLD(4,4,RD(33))/10.*FLD(0,4,RD(32))/100.
92      S *FLD(4,4,RD(32))/1000.
93      C
94      RETURN
95      END

```

BPRT,S V.MOVE

```

1  SA*LIB(1)-MOVE
2  SUBROUTINE MOVE (FROM,IFBYTE,TO,IYBYTE,NBYTES)
3  C
4  C THIS ROUTINE MOVES BYTES FROM AN INPUT ARRAY
5  C FROM TO AN OUTPUT ARRAY TO IFBYTE IS
6  C THE BYTE NUMBER OF THE FIRST BYTE TO BE TRANSFERRED
7  C FROM THE INPUT ARRAY. IYBYTE IS THE BYTE NUMBER
8  C OF THE POSITION IN THE OUTPUT ARRAY WHICH WILL
9  C RECEIVE THE TRANSFERRED BYTE. NBYTES IS THE
10 C NUMBER OF BYTES TO BE TRANSFERRED.
11 C
12 C DIMENSION FROM(1),TO(1),MASK(10),ISM(9)
13 C DATA ISM/28,20,12,4,0,24,16,8,0/
14 C DATA MASK/268435455,-267386680,-1044480,-4080,-15,
15 C -4278190080,-16711680,-65280,
16 C -255,4294967295/
17 C DO 1000 NM=1,NBYTES
18 C.....GET A BYTE
19 C
20 C NB=(NM-1)+IFBYTE
21 C IN=(NB-1)*8/36+1
22 C IF((NB-5)/9*9.EQ.(NB-5)) GO TO 100
23 C IBS=(NB-1)*8-(IN-1)*36
24 C IF (IBS.EQ.0) IBS=0
25 C IND=FLO (IBS,8,FROM(IN))
26 C GO TO 200
27 100 IND=FLO (32,4,FROM(IN))+16+FLO(0,4,FROM(IN+1))
28 200 CONTINUE
29 C
30 C.....PUT A BYTE
31 C
32 C NB=(NM-1)+IYBYTE
33 C IBS=MOD (NB-1,9)+1
34 C NEL=MOD (NB-1,9)+1
35 C IN=(NB-1)*8/36+1
36 C IF ((NB-5)/9*9.EQ.(NB-5)) GO TO 300
37 C IND=IND*2**ISM(IPS)
38 C TO(IN)=AND(TO(IN),MASK(NEL))
39 C TO(IN)=OR(TO(IN),IND)
40 C GO TO 400
41 300 TO(IN)=AND(TO(IN),MASK(5))
42 C TO(IN)=OR(TO(IN),FLO(28,4,IND))
43 C TO(IN+1)=AND(TO(IN+1),MASK(10))
44 C TO (IN+1)=OR(TO(IN+1),FLO(32,4,IND)*2**32)
45 400 CONTINUE
46 C
47 1000 CONTINUE
48 RETURN
49 END

```

*PRT,S V.MAPTRN

SA*LIB(1).CONVERSION

```

1  C MAIN ROUTINE FOR CONVERSION PROGRAM.
2  C
3  C THIS PROGRAM CONVERTS THE VELOCITY,
4  C ACCELERATION,AND MAGNETIC VECTORS
5  C FROM INSTRUMENT TO GEOMAGNETIC COORDINATES.
6  C
7  C *****
8  C

```

```

9      C COMMONS FOR ZREAD
10     COMMON/RHDR/LR,NR,NBR,NHBR,NHFR,NFR,NIR,NAR,IPR(15)
11     COMMON/RDATA/VR(15,500)
12     COMMON/RDOCF/FDOCR(40)
13     COMMON/RDOCI/IDOCR(20)
14     COMMON/RDOCA/ADOCR(50)
15     C
16     COMMON/DIAGS/MSGR,MSGU,NNNR,NNNU,NNIP,NNF,NNI,NNA,TRST,IVST
17     C
18     C INITIALIZE CONTROL HEADER
19     DATA LR,NR,NFR,NIR,NAR/15,500,40,20,50/
20     DATA NNNR,NNIP,NNF,NNI,NNA/500,15,40,20,50/
21     MSGR=0
22     MSGU=0
23     C
24     C *****
25     C
26     C COMMONS FOR CONVERSION
27     COMMON/BETA/BHAT(3,3),ORTHOG(4,3),ACCAL(3)
28     COMMON/NCYCLE,ACCL(3),XNAG(3),U(3),A(3),M(3)
29     C
30     C INITIALIZE VELOCITY ORTHOGONALIZATION MATRIX
31     DATA ORTHOG(1,1),ORTHOG(2,1),ORTHOG(3,1),ORTHOG(4,1)/
32     S -.055,-.803,1.58,-.794/
33     DATA ORTHOG(1,2),ORTHOG(2,2),ORTHOG(3,2),ORTHOG(4,2)/
34     S -.197,-.684,-.011,-.759/
35     DATA ORTHOG(1,3),ORTHOG(2,3),ORTHOG(3,3),ORTHOG(4,3)/
36     S .612,.849,-.071,.659/
37     C
38     C INITIALIZE ACCELEROMETER CALIBRATION COEFFICIENTS
39     DATA ACCAL(1),ACCAL(2),ACCAL(3)/-1.000,1.000,1.000/
40     C
41     C *****
42     C
43     10 CONTINUE
44     CALL ZREAD(10,IF,0)
45     C
46     C CHANCE VARIABLE NAMES
47     IPR(5)='VLOGG1'
48     IPR(6)='VLOGG2'
49     IPR(7)='VLOGG3'
50     IPR(8)='MAGG1'
51     IPR(9)='MAGG2'
52     IPR(10)='MAGG3'
53     IPR(11)='ACCLG1'
54     IPR(12)='ACCLG2'
55     IPR(13)='ACCLG3'
56     C
57     C ADD TO ADOC
58     ADOCR(37)='GEOMA'
59     ADOCR(38)='G COOP'
60     ADOCR(39)='D'
61     C
62     NCYCLE=0
63     C
64     20 CONTINUE
65     NCYCLE=NCYCLE+1
66     IF (NCYCLE.GT.NR) GO TO 100
67     C
68     C
69     50 CONTINUE
70     C
71     C CALL VECTOR CONVERSION SUBROUTINES

```

```

72          CALL VCTCON
73          C
74          C *****
75          C WRITE NEW SEGMENT
76          C
77          100 CONTINUE
78          IF (NCYCLE.LE.NR) GO TO 20
79          CALL ZWRIT (-20,IF,G)
80          C
81          C CHECK FOR END OF CAST
82          IF (IDOCR(1).NE.1) GO TO 10
83          C
84          C *****
85          END

```

BPRT,S V.CONVVECTOR

3A*LIB(1).CONVVECTOR

```

1          SUBROUTINE VCTCON
2          C
3          C THIS SUBROUTINE CONVERTS THE VELOCITY, ACCELERATION,
4          C AND MAGNETIC VECTORS FROM INSTRUMENT COORDINATES
5          C INTO GEOMAGNETIC COORDINATES.
6          C
7          C *****
8          C COMMONS FOR CONVERSION
9          COMMON/BETA/RMAT(3,3),ORTHOG(4,3),ACCAL(3)
10         COMMON/RDATA/VR(15,500)
11         COMMON/RDOCF/FDOCR(40)
12         COMMON/RDOCI/IDOCR(20)
13         C
14         COMMON/NCYCLE,ACC(3),XMAG(3),U(3),A(3),H(3)
15         REAL W(3),LENGTH
16         C
17         C *****
18         C
19         C
20         C *****
21         C
22         C CALL SUBROUTINE TO CREATE TRANSFORM MATRIX
23         CALL TRNMTX
24         C
25         C *****
26         C
27         C DO COORDINATE CONVERSIONS
28         DO 30 I=1,3
29             U(I)=0
30             A(I)=0
31             H(I)=0
32         DO 30 J=1,3
33             U(I)=U(I)+BMAT(I,J)*VR(J+4,NCYCLE)
34             A(I)=A(I)+BMAT(I,J)*VR(J+10,NCYCLE)
35             H(I)=H(I)+BMAT(I,J)*VR(J+7,NCYCLE)
36         WRITE(6,2001) U(1),U(2),U(3)
37         2001 FORMAT(' V - AFTER :',3G20.7)
38         LENGTH=SQRT(U(1)*U(1)+U(2)*U(2)+U(3)*U(3))
39         WRITE(6,1001) LENGTH
40         1001 FORMAT(' ABSOLUTE VELOCITY AFTER TRANSFORM =',6G20.5//)
41         C
42         C PUT NEW VALUES INTO DATA ARRAY
43         DO 50 I=1,3
44             VR(I+4,NCYCLE)=U(I)

```

```

45      VR(I+10,NCYCLE)=A(I)
46      50      VR(I+7,NCYCLE)=H(I)
47      C
48      C *****
49      RETURN
50      END

```

BPRT,S V.TRANSFORMS

3A*LIB(1).TRANSFORMS

```

1      SUBROUTINE TRNMTX
2      C
3      C THIS SUBROUTINE CALIBRATES THE ACCELERATION VECTOR,
4      C NORMALIZES THE ACCELERATION AND MAGNETIC VECTORS,
5      C AND CREATES THE TRANSFORM MATRIX 'BMAT'.
6      C
7      C *****
8      C
9      C COMMONS FOR CONVERSION
10     COMMON/RODATA/VR(15,500)
11     COMMON/KDOCF/FDOCF(40)
12     COMMON/BETA/BMAT(3,3),ORTHOG(4,3),ACCAL(3)
13     C
14     COMMON NCYCLE,ACC(3),XMAG(3),U(3),A(3),H(3)
15     C
16     C *****
17     C
18     C CALIBRATE ACCELERATION VECTOR
19     DO 10 I=1,3
20     10      VR(I+10,NCYCLE)=ACCAL(I)*VR(I+10,NCYCLE)
21     C
22     C *****
23     C
24     C NORMALIZE 'A' AND 'H' VECTORS
25     XNA=0
26     XNH=0
27     DO 20 I=1,3
28     20      XNA=XNA+VR(I+10,NCYCLE)**2
29     20      XNH=XNH+VR(I+7,NCYCLE)**2
30     C
31     XNA=SQRT(XNA)
32     XNH=SQRT(XNH)
33     C
34     DO 30 I=1,3
35     30      ACC(I)=VR(I+10,NCYCLE)/XNA
36     30      XMAG(I)=VR(I+7,NCYCLE)/XNH
37     C
38     C CALCULATE DIP FROM 'A' AND 'H'
39     SINDIP=0
40     DO 40 I=1,3
41     40      SINDIP=SINDIP+ACC(I)*XMAG(I)
42     DIP=ASIN(SINDIP)
43     COSDIP=COS(DIP)
44     C
45     C *****
46     C
47     C CREATE TRANSFORM MATRIX
48     DO 50 I=1,3
49     50      BMAT(3,I)=-ACC(I)
50     C
51     DO 70 I=1,3
52     70      BMAT(2,I)=(XMAG(I)+SINDIP*BMAT(3,I))/COSDIP

```

```

53      C
54      XT2=1.0-BMAT(2,1)**2-BMAT(3,1)**2
55      IF (XT2.LT.0) XT2=-XT2
56      PMAT(1,1)=SQRT(XT2)
57      IF (XMAG(2).LT.0) BMAT(1,1)=-BMAT(1,1)
58      C
59      XT=ACC(2)*XMAG(3)-ACC(3)*XMAG(2)
60      BMAT(1,2)=-BMAT(1,1)*(ACC(1)*XMAG(3)-ACC(3)*XMAG(1))/XT
61      BMAT(1,3)=BMAT(1,1)*(ACC(1)*XMAG(2)-ACC(2)*XMAG(1))/XT
62      C
63      C *****
64      C
65      RETURN
66      END

```

BPRT,S V.MAPCON

JA*LIB(1)-PREFIX

```

1      C*****
2      C PROGRAM V.PREFIX
3      C PURPOSE : TO 1. COMPUTE WI FROM PRESSURE DERIVATIVES AND
4      C 2. ATTEMPT TO FIX EFFECTS OF OVERRA .ING
5      C
6      C*****
7      C
8      C
9      C
10     C
11     COMMON /KSUFF/ KSEG,KNR,KREAL
12     COMMON/BETA/UNORTH(4,3),ORTHOG(4,3),W(3),Q(3)
13     C COMMONS FOR ZREAD
14     COMMON/RHDR/LR,NR,NFR,NBR,NMFR,NFR,NIR,NAR,IPR(15)
15     COMMON/RDATA/VR(15,500)
16     COMMON/RDOCF/FDOCR(40)
17     COMMON/RDOC I/IDOCR(20)
18     COMMON/RDOCA/ADOCR(50)
19     C
20     COMMON/DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
21     C
22     C COMMONS FOR ZWRIT
23     C
24     COMMON/WHDR/LW,NW,NBW,NMBW,NMFW,NFW,NIW,NAW,IPW(15)
25     COMMON/WDATA/VW(15,500)
26     COMMON/WDOCF/FDOCW(40)
27     COMMON/WDOC I/IDOCW(20)
28     COMMON/WDOCA/ADOCW(50)
29     C
30     C
31     REAL VROT(5000)
32     C
33     C*****
34     C INITIALIZE COMMONS FOR ZREAD AND ZWRITE
35     C*****
36     C
37     DATA LR,NR,NFR,NIR,NAR/15,500,40,20,50/
38     DATA LW,NW,NFW,NIW,NAW/15,500,40,20,50/
39     DATA NNNR,NNNW,NNIP,NNF,NNI,NNA/500,500,15,40,20,50/
40     C
41     C*****
42     C INITIALIZE MATRIX FOR CONVERSION BACK TO UNORTHOGONAL ACOUSTIC
43     C AXES COORDINATES.
44     C*****

```



```

108      7000      CONTINUE
109              DO 7001 IP=1,20
110      7001      ILOCW(IP)=IDOCF(IP)
111              DO 7002 IP= 1,50
112      7002      ADOCW(IP)=ADOCF(IP)
113              DO 7003 IP=1,15
114      7003      IPW(IP)=IPR(IP)
115              IPW(14) = 'WI'
116              NMHW = NMWR
117              NMFW = NMFR
118      C
119              NPOINT = NSEG1*500      MAXIMUM POSS. NO. OF DATA CYCLES
120      C
121      C
122      C*****
123      C*****
124      C***
125      C***      BEGIN THE MAIN LOOPS      ***
126      C***
127      C*****
128      C*****
129      C
130      C
131              DO 1000 I = 1 , NSEG1
132              DO 3000 II= 1,500
133              DO 3000 JJ = 1,15
134              VW(JJ,II) = VR(JJ,II)
135      3000      CONTINUE
136              DO 1001 J = 1, 500
137              ICYC = (I-1)*500+J
138              IF( ICYC .LE. NN*1) IPTR= ICYC
139              IF( ICYC .GT. NN*1) IPTR= NN*1
140      C
141      C*****
142      C      COMPUTE WI AT THE I-TH POINT ( APPROXIMATELY) PLACE IN *
143      C      VARIABLE 14 IN THE OUTPUT BUFFER      *
144      C*****
145      C
146              DPDT = 0
147              DO 2000 K = 1,NN
148              LP=NN+1-K
149              LM =NN+1-K
150              DPDT=DPDT+K*(VRX(1,LP)-VRX(1,LM))
151      2000      CONTINUE
152              DPDT=FACTOR*DPDT
153              VW(14,J) = DPDT
154      C
155      C*****
156      C      TEST FOR GROSS INEQUALITY OF WI AND W. ATTEMPT TO *
157      C      CORRECT W BY      *
158      C      1. RACK TRANSFORMING U,V,W TO Q1,Q2,Q3      *
159      C      2. REPLACING Q1 AND Q3 BY .657WI-.534 AND      *
160      C      .60185WI-.227, RESPECTIVELY AND      *
161      C      3. RETRANSFORMING THE Q COMPONENTS TO U,V,AND W*
162      C*****
163      C
164              IF( ABS(DPDT-VRX(7,IPTR)).LT.42)
165              1      GOTO 5000      2 TEST FOR PAD VERT.VEL.
166      C*****
167      C* CONVERT VELOCITIES BACK TO ORIGINAL ACOUSTIC COORDS      *
168      C*****
169      C
170              DO 2500 K=1,3

```

```

171      W(K) = VRX(K+4,IPR)
172      2500      CONTINUE
173      DO 2600 K = 1,3
174      V = -UNORTH(1,K)
175      DO 2601 JJ = 1,3
176      V = V+W(JJ)*UNORTH(JJ+1,K)
177      2601      CONTINUE
178      Q(K) = V
179      2600      CONTINUE
180      C
181      Q(1) = 0.658*DPDT+ 0.634 & CORRECT 1-ST ACOUSTIC AXIS
182      Q(3) = 0.602*DPDT+ 0.227 & CORRECT 3-RD ACOUSTIC AXIS
183      C
184      C*****
185      C-RETRANSFORM BACK TO ORTHOGONAL COORDINATES
186      C*****
187      C
188      DO 2610 KK= 1,3
189      W(KK) = ORTHOG(1,KK)
190      DO 2611 LL= 1,3
191      W(KK) = W(KK)+ Q(LL)*ORTHOG(LL+1,KK)
192      2611      CONTINUE
193      W(KK+4,J) = W(KK)
194      2610      CONTINUE
195      5000      CONTINUE
196      IF(ICYC.GT. NM .OR. I.GT. 1) CALL STEPX(S1,S2,1)
197      2      CONTINUE
198      1001      CONTINUE
199      IF(1.EQ.NSEGL)IDOCW(1)=1 & SET END CAST FLAG ON LAST SEG
200      CALL ZWRIT(NUOUT,IF,0) & WRITE SEGMENT TO OUTPUT FILE
201      1000      CONTINUE
202      STOP
203      1      CONTINUE
204      C
205      IDOCW(1) = 1
206      CALL ZWRIT(NUOUT,IF,0)
207      STOP
208      END

```

@PRT,S.V.2BUFF

ODE*31*FCHFILE1,1).VFIX1

```

      C      THIS PROGRAM COMPUTES THE PROFILER CORRECTED VERTICAL
      C      VELOCITY USING INTEGRATED ACCELERATIONS
      3      C
      4      C      BEFORE EXECUTING, THE INPUT AND OUTPUT FEB FILES
      5      C      MUST BE ASSIGNED TO SOME UNIT NUMBERS
      6      C
      7      C      MAPPING ELEMENT: FCHFILE1.MVFIX1
      8      C
      9      DIMENSION A(4,5),X(4),W(500),AIJ1(4,5),DW(500),DWTIME(500)
      10     COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(20)
      11     COMMON /DATA/VP(10000)
      12     COMMON /RDOCF/FDOCR(40)
      13     COMMON /RDOC1/IDOCR(20)
      14     COMMON /RDOCA/ADOCR(200)
      15     COMMON /DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
      16     COMMON /DATA/A,T1,NUMT,NUMP,NUMV,ISTART,TLAST,ALAST,PRESS1,WI,F1,X
      17     COMMON /DATA1/AIJ1,WSAVE,TSAVE,ASAVE,TINDIF,INF,X,GRAY,IFIRST,NUMA
      18     COMMON /DATA2/WCHG,DW,DWTIME,IDW,F2,WRAR,NSSEG,.NDEXS,IUNR,IUNW,W
      19     DATA LR,NR,NFR,NIR,NAR/20,500,40,20,50/
      20     DATA NMBR,NNNW,NNIP,NNF,NNI,NNA/500,500,20,40,20,200/

```

```

21      C
22      GRAV=0.
23      ACORR=9.99
24      RMOG1=.9945
25      C
26      C      GAP= MAXIMUM TIME (SECONDS) BEFORE SYSTEM IS SOLVED
27      C      AND CORRECTED VALUES ARE COMPUTED
28      C      GAP1= MAXIMUM TIME GAP (SECONDS) BEFORE LINEAR
29      C      INTERPOLATION FOR MISSING DATA IS PERFORMED
30      C
31      GAP=1.
32      GAP1=0.1
33      C
34      IFIRST=0
35      C
36      WRITE(6,1)
37      1  FORMAT(' INPUT UNIT NUMBER, NUMBER OF SEGMENTS, STARTING SEGMENT,
38      MESSAGE LEVEL',
39      READ(5,2)IUNR,NSEG,NSSEG,MSGR
40      2  FORMAT('
41      WRITE(6,3)
42      3  FORMAT(' OUTPUT UNIT NUMBER, MESSAGE LEVEL')
43      READ(5,2)IUNW,MSGW
44      C
45      NINDEX=0
46      IBGN=NSSEG
47      DO 100 ISEG=1,NSSEG
48      4  CALL ZREAD(IUNR,IF,NSSEG)
49      IF(IF.NE.0)GO TO 99
50      IFLAG=0
51      NNSSEG=NSSEG
52      C
53      C      LOOP DETERMINES VARIABLE LOCATIONS
54      C
55      DO 5 J=1,LR
56      IF(IPR(J).EQ.'VLOC13')NUMV=J
57      IF(IPR(J).EQ.'ACCL13')NUMA=J
58      IF(IPR(J).EQ.'PRESS')NUMP=J
59      IF(IPR(J).EQ.'RELSEC')NUMT=J
60      5  CONTINUE
61      C
62      INDEX=NINDEX
63      8  ISTART=INDEX+1
64      C
65      C      CALL SUBROUTINE TO INITIALIZE STARTING VALUES FOR THE
66      C      FIRST READ OR FOR THE FIRST READ AFTER A TIME GAP
67      C
68      IF(IFIRST.EQ.0)CALL INIT
69      C
70      DO 10 I=ISTART,NR
71      IND=(I-1)*LR
72      TIMDIF=VR(IND+NUMT)-TLAST
73      C
74      C      IF TIME DIFFERENCE TOO LARGE, SOLVE THE SYSTEM
75      C
76      IF(TIMDIF.GT.GAP)GO TO 11
77      C
78      INDEX=INDEX+1
79      ACC=VR(IND+NUMA)+ACORR
80      C
81      C      IF DATA VALUES ARE MISSING, PERFORM INTERPOLATION
82      C
      IF(TIMDIF.GE.GAP1)CALL CORREC(IND,VLAST,GAP)

```

```

      C
85      C      IF NO DATA VALUES MISSING, PERFORM ACCELERATION
86      C      INTEGRATION (F1)
87      C
88      C      IF (TINDIF.LT.GAP1) F1=F1+(VR(IND+NUMA)+ACORR+ALAST)*TINDIF/2.
89      C
90      C      SAVE LAST VALUES FOR NEXT CYCLE
91      C
92      C      TLAST=VR(IND+NUMT)
93      C      ALAST=VR(IND+NUMA)+ACORR
94      C      MLAST=VR(IND+NUMV)
95      C      PLAST=VR(IND+NUMP)
96      C
97      C      T2= TOTAL TIME
98      C      W= MEASURED VELOCITY (CONVERTED TO METERS/SEC)
99      C      WBAR= AVERAGE VELOCITY (BASED ON PRESSURE CHANGE)
100     C
101     C      F2=TLAST-T1
102     C      W=GRAV+F2+VR(IND+NUMV)/100
103     C      WBAR=0.
104     C      IF (F2.NE.0.) WBAR=-(PRESS1-VR(IND+NUMP))/RHOG1/F2
105     C
106     C      SUM NEW VALUES TO ARRAYS IF NOT ALREADY DONE THROUGH
107     C      THE CORREC SUBROUTINE
108     C
109     C      IF (TINDIF.LT.GAP1) CALL ARRAYS
110     C
111     C      10      CONTINUE
112     C      GO TO 90
113     C
114     C      PROGRAM CONTINUES HERE IF A TIME GAP IS DISCOVERED
115     C
116     C      11      NINDEX=INDEX
117     C
118     C      TAKE CARE OF THE CASE WHERE A TIME GAP OCCURS BETWEEN SEGMENTS
119     C
120     C      IF (INDEX.EQ.0) NNSSEG=NNSSEG-1
121     C      IF (INDEX.EQ.0) NINDEX=NR
122     C
123     C      CALL SUBROUTINES TO SOLVE THE SYSTEM AND COMPUTE CORRECTIONS
124     C
125     C      CALL SOLVE(PLAST)
126     C      CALL WICOMP(ACORR,NINDEX,NNSSEG,IBGN,IFLAG,LAP1)
127     C
128     C      90      NSSEG=NSSEG+1
129     C
130     C      IF SYSTEM HAS NOT BEEN SOLVED, RESET INDEX AND SET FLAG
131     C      FOR VARIABLE INITIATION
132     C
133     C      IF (IFLAG.EQ.0) NINDEX=0
134     C      IF (IFLAG.EQ.0) IFIRST=1
135     C
136     C      IF A COMPLETE SEGMENT HAS NOT BEEN READ, RETURN TO READ
137     C      WITHOUT INCREMENTING THE LOOP COUNTER
138     C
139     C      IF (INDEX.LT.NR) GO TO 4
140     C
141     C      100     CONTINUE
142     C
143     C      SOLVE SYSTEM AND COMPUTE CORRECTIONS AFTER LAST READ
144     C
145     C      NINDEX=INDEX
146     C      NNSSEG=NSSEG-1

```

```

147      CALL SOLVE(PLAST)
148      CALL WICOMP(ACORR,NINDEX,MNSSEG,IBEN,IFLAG,GAP)
149      GO TO 999
150      99  WRITE(6,7)
151      7   FORMAT(/,' READ ERROR')
152      999 END

```

ODE*11*FCHFILE1(1).INIT/VFIX1

```

      C   THIS SUBROUTINE INITIATES DATA FOR THE BEGINNING OF THE
      C   PROGRAM OR AFTER A TIME GAP HAS OCCURRED
      C
3      C
4      SUBROUTINE INIT
5      DIMENSION A(4,5),VI(500),AIJ1(4,5),X(4),DV(500),DWTIME(500)
6      COMMON /RDATA/VR(10000)
7      COMMON /RHDR/LR,NR,NBR,NHBR,NHFR,NFR,NIR,NAR,IPR(20)
8      COMMON /DATA/A,T1,NUMT,NUMP,NUMV,ISTART,TLAST,ALAST,PRESS1,WI,F1,X
9      COMMON /DATA1/AIJ1,MSAVE,TSAVE,ASAVE,TINDIF,INDEX,GRAY,IFIRST,NUMA
10     COMMON /DATA2/MCHG,DW,DWTIME,IDW,F2,WRAR,NSSEG,INDEXS,IUNR,IUNW,M
11     C
12     INDEXS=INDEX+1
13     IDW=0
14     IND=(ISTART-1)*LR
15     C
16     C   T1,PRESS1= START TIME AND PRESSURE FOR INITIAL READ OR AFTER GAP
17     C
18     T1=VR(IND+NUMT)
19     TLAST=T1
20     PRESS1=VR(IND+NUMP)
21     F1=0.
22     MCHG=0.
23     ALAST=0.
24     MSAVE=0.
25     TSAVE=T1
26     ASAVE=0.
27     C
28     C   LOOP ZEROES ARRAY VALUES
29     C
30     DO 10 I=1,4
31       DO 10 J=1,5
32         AIJ1(I,J)=0.
33       10  A(I,J)=0.
34     RETURN
35     END

```

ODE*11*FCHFILE1(1).CORREC/VFIX1

```

      C   THIS SUBROUTINE INTERPOLATES AND FILLS GAPS FOR MISSING DATA
      C
      C
3      SUBROUTINE CORREC(IND,ALAST,GAP)
4      COMMON /RHDR/LR,NR,NBR,NHBR,NHFR,NFR,NIR,NAR,IPR(20)
5      COMMON /RDATA/VR(10000)
6      COMMON /DATA/A,T1,NUMT,NUMP,NUMV,ISTART,TLAST,ALAST,PRESS1,WI,F1,X
7      COMMON /DATA1/AIJ1,MSAVE,TSAVE,ASAVE,TINDIF,INDEX,GRAY,IFIRST,NUMA
8      COMMON /DATA2/MCHG,DW,DWTIME,IDW,F2,WRAR,NSSEG,INDEXS,IUNR,IUNW,M
9      DIMENSION A(4,5),VI(500),AIJ1(4,5),X(4),DV(500),DWTIME(500)
10     C
11     DW1=VR(IND+NUMV)-ALAST
12     SLOPE=DW1/TINDIF
13     C
14     C   COMPUTE THE MAXIMUM NUMBER OF TIMES TO LOOP IN ORDER TO DETERMINE THE
15     C   NUMBER OF DATA POINTS MISSING (BASED ON AN OPTIMUM
16     C   TIME DIFFERENCE OF 0.0625 SECONDS)
17     C
18     LOOP=INT(GAP/0.0625)+1

```

```

19      CHKPTS=0
20      C
21      C      LOOP DETERMINES THE NUMBER OF POINTS MISSING
22      C
23      DO 10 I=1,LOOP
24      PTS=ABS(TIMDIFF/I-0.0625)
25      IF (PTS.LE.CHKPTS) NPTS=I
          10      CHKPTS=PTS
          C
26      C      DIVIDE THE TIME INCREMENT EQUALLY AND COMPUTE THE
27      C      VELOCITY INCREASE PER INCREMENT
28      C
29      TIMDIFF=TIMDIFF/NPTS
30      DW2=SLOPE*TIMDIFF
31      C
32      C      LOOP FINDS INTERPOLATED VALUES OF F1, W, F2, AND SUMS TO ARRAYS
33      C
34      DO 20 I=1,NPTS
35      F1=F1+DW2/100
36      W=(WLAST+DW2*I)/100
37      F2=F2+TIMDIFF
38      CALL ARRAYS
39      CONTINUE
          20      C
40      C      RESET TIME DIFFERENCE
41      C
42      TIMDIFF=TIMDIFF*NPTS
43      RETURN
44      END

```

DDF₁*FCHFILE(1).ARRAYS/VFIX1

```

      C      THIS SUBROUTINE COMPUTES VALUES IN THE MATRIX USED TO
      C      SOLVE FOR ALPHA, BETA, GAMMA, AND LAMBDA
      C
3      SUBROUTINE ARRAYS
4      DIMENSION A(4,5),W(500),AIJ1(4,5),AIJ2(4,5),DW(500),DWTIME(500)
5      DIMENSION X(4)
6      COMMON /DATA/A,T1,NUMT,NUMP,NUMV,ISTART,TLAST,ALAST,PRESS1,WI,F1,X
7      COMMON /DATA1/AIJ1,WSAVE,TSAVE,ASAVE,TIMDIFF,INDEX,GRAV,IFIRST,NUMA
8      COMMON /DATA2/WCHG,DW,DWTIME,IDW,F2,WBAR,NSSEG,INDEXS,IUNR,IUNW,W
9      C
10     C      ARRAY AIJ2 STORES VALUES FOR THIS DATA POINT
11     C      ARRAY AIJ1 STORES VALUES FOR THE LAST DATA POINT
12     C
13     AIJ2(1,1)=F1*F1
14     AIJ2(1,2)=F1
15     AIJ2(1,3)=-F1*F2
16     AIJ2(1,4)=F1/2
17     AIJ2(1,5)=F1*W
18     AIJ2(2,1)=F1
19     AIJ2(2,2)=1.
20     AIJ2(2,3)=-F2
21     AIJ2(2,4)=0.5
22     AIJ2(2,5)=W
23     AIJ2(3,1)=F1*F2
24     AIJ2(3,2)=F2
25     AIJ2(3,3)=-F2*F2
26     AIJ2(3,4)=F2/2
27     AIJ2(3,5)=F2*W
28     AIJ2(4,1)=F1
29     AIJ2(4,2)=1.
30     AIJ2(4,3)=-F2
31     AIJ2(4,4)=0.

```

```

33      AIJ2(4,5)=WBAR*GRAV*F2
34      C
35      C      LOOP INTEGRATES AIJ2 VALUES INTO ARRAY A, AND STORES
36      C      AIJ2 VALUES INTO ARRAY AIJ1
37      C
38      DO 10 I=1,4
39      DO 10 J=1,5
40      AI(I,J)=A(I,J)+(AIJ1(I,J)+AIJ2(I,J))*TIMDIF/2
41      AIJ1(I,J)=AIJ2(I,J)
42      CONTINUE
43      A(4,5)=WBAR*F2
44      RETURN
45      END

```

FRT,S FCHFILE1,SOLVE/VFIX1,.WICOMP/VFIX1,.WRITE/VFIX1

CODE331=FCHFILE1(1).SOLVE/VFIX1

```

      C      THIS SUBROUTINE SOLVES THE MATRIX EQUATION AND OUTPUTS
      C      START TIMES AND END TIMES FOR TIME AND PRESSURE FOR
      C      THE PERIOD SINCE THE LAST TIME GAP. ALSO OUTPUT ARE
      C      ALPHA, BETA, GAMMA, AND LAMBDA
3      C
4      C
5      C
6      SUBROUTINE SOLVE(PLAST)
7      DIMENSION AA(4,5),A(4,5),X(4),WI(500),AIJ1(4,5)
8      COMMON /RHDR/LR,NR,NBR,NHBR,NHFR,NFR,NIR,NAR,IPR(20)
9      COMMON /RDATA/VR(10000)
10     COMMON /DATA/A,T1,NUMT,NUMP,NUMV,ISTART,TLAST,ALAST,PRESS1,WI,F1,X
11     COMMON /DATA1/AIJ1,WSAVE,TSAVE,ASAVE,TIMDIF,INDEX,GRAV,IFIRST,NJWA
12     C
13     WRITE(6,51)T1,TLAST,PRESS1,PLAST
14     51  FORMAT(' START TIME, END TIME',2F20.4,/,
15     & ' START PRESSURE, END PRESSURE',2F20.4,/)
16     DO 20 I=1,4
17     DO 20 J=1,5
18     20  AA(I,J)=A(I,J)
19     XMAT=SIMUL(4,AA,X,0,0,4)
20     WRITE(6,14)XMAT
21     14  FORMAT(' XMAT =',610.5)
22     WRITE(6,6)X(1),X(2),X(3),X(4)
23     6   FORMAT(' ALPHA =',610.5,3X,' BETA =',610.5,3X,' GAMMA =',610.5,
24     & 3X,' LAMBDA =',610.5)
25     RETURN
26     END

```

CODE331=FCHFILE1(1).WICOMP/VFIX1

```

      C      THIS SUBROUTINE COMPUTES THE CORRECTIONS TO VELOCITY
      C
3      C
4      SUBROUTINE WICOMP(ACORR,NINDEX,NNSSEG,IBGN,FLAG,GAP1)
5      DIMENSION A(4,5),X(4),WI(500),AIJ1(4,5),DW(500),DUTIME(500)
6      COMMON /RHDR/LR,NR,NBR,NHBR,NHFR,NFR,NIR,NAR,IPR(20)
7      COMMON /RDATA/VR(10000)
8      COMMON /DATA/A,T1,NUMT,NUMP,NUMV,ISTART,TLAST,ALAST,PRESS1,WI,F1,X
9      COMMON /DATA1/AIJ1,WSAVE,TSAVE,ASAVE,TIMDIF,INDEX,G' V,IFIRST,NJWA
10     COMMON /DATA2/WCHG,DW,DUTIME,IDW,F2,WBAR,NSSEG,INDEXS,IUNR,IUNW,W
11     C
12     ISTART=INDEXS
13     IFIRST=0
14     IEND=NNSSEG-IBGN+1
15     DO 10 IWRITE=1,IEND
16     C
17     C      READ SEGMENTS BEGINNING WITH START SEGMENT OR THE LAST
      C      SEGMENT WHERE A TIME GAP OCCURRED

```

```

18      C
19      CALL ZREAD(IUNR,IF,IBGN)
20      C
21      SET LAST CYCLE TO READ
22      C
23      INDEX1=NR-1
24      IF(IWRITE.EQ.IEND)INDEX1=NINDEX-1
25      C
26      IND=(ISTART-1)*LR
27      CHGTIM=VR(IND+NUMT)-TSAVE
28      C
29      COMPUTE INSTRUMENT VELOCITY (WI) FOR THE FIRST CYCLE
30      C      (WI VALUES ARE COMPUTED BY EITHER INTEGRATION OF
31      C      ACCELERATION OR BY CHANGE IN VELOCITY, DEPENDING
32      C      ON THE TIME GAP)
33      C
34      IF(CHGTIM.GE.GAP1)WI(ISTART)=WSAVE*(X(1)*(VR(IND+NUMV)-WLASTS)/
35      # 100-X(3)*CHGTIM
36      IF(CHGTIM.LT.GAP1)WI(ISTART)=WSAVE*(X(1)*(ASAVE+VR(IND+
37      # NUMA)+2*ACORR)-2*(GRAV*X(3)))*CHGTIM/2
38      C
39      FOR FIRST CYCLE, WI=BETA
40      C
41      IF(IFIRST.EQ.0)WI(ISTART)=X(2)
42      C
43      COMPUTE WI VALUES TO END OF SEGMENT OR TO LAST CYCLE READ
44      C
45      DO 30 I=ISTART,INDEX1
46      IND=(I-1)*LR
47      CHGTIM=VR(IND+LR+NUMT)-VR(IND+NUMT)
48      IF(CHGTIM.GE.GAP1)WI(I+1)=WI(I)+X(1)*(VR(IND+LR+NUMV)-
49      # VR(IND+NUMV))/100-X(3)*CHGTIM
50      IF(CHGTIM.LT.GAP1)WI(I+1)=WI(I)+(X(1)*(VR(IND+NUMA)+
51      # VR(IND+LR+NUMA)+2*ACORR)-2*(GRAV*X(3)))*CHGTIM/2
52      30 CONTINUE
53      C
54      INDEX2=INDEX1+1
55      C
56      CALL SUBROUTINE TO WRITE CORRECTED VELOCITY TO OUTPUT FILE
57      C
58      CALL WRITE(IF,WLASTS,INDEX2)
59      C
60      RESET VALUES FOR READING NEXT SEGMENT
61      C
62      IBGN=IBGN+1
63      IFIRST=1
64      ISTART=1
65      10 CONTINUE
66      C
67      SET VALUES FOR RETURN TO MAIN PROGRAM
68      C
69      IF(INDEX.NE.0)NSSEG=NNSSEG-1
70      IF(INDEX.NE.0)IBGN=IBGN-1
71      IFLAG=1
72      IFIRST=0
73      IF(INDEX.EQ.0)NINDEX=0
74      RETURN
75      END

```

```

ODF=310FCHFILE1(1).WRITE/VFIX1

```

```

      C      THIS SUBROUTINE WRITES CORRECTED VELOCITY VALUES TO A NEW FEB FILE
      C      (THE OUTPUT FILE CONTAINS 2 VARIABLES MORE THAN THE INPUT FILE)
      C
      3

```



```

67      C
68      C      SAVE VALUES FOR RETURN TO SUBROUTINE WICOMP
69      C
70      WSAVE=WI(NR)
71      IND2=(NR-1)*LR
72      TSAVE=VR(IND2+NUMT)
73      ASAVE=VR(IND2+NUMA)
74      WLASTS=VR(IND2+NUMV)
75      C
76      C      CALL ZWRIT ONLY IF THE ENTIRE SEGMENT HAS BEEN COMPLETED
77      C
78      IF(INDEX2.LT.NR)RETURN
79      CALL ZWRIT(IUNW,IF,0)
80      RETURN
81      END

```

PK FCHFILE,SIMUL

ODF331,FCHFILE(1),SIMUL

```

      C      FUNCTION SIMUL (N,A,X,EPS,INDIC,NRC)
      C
      C      WHEN INDIC IS NEGATIVE, SIMUL COMPUTES THE INVERSE OF THE N BY N
      C      MATRIX A IN PLACE. WHEN INDIC IS ZERO, SIMUL COMPUTES THE N
      C      SOLUTIONS X(1)...X(N) CORRESPONDING TO THE SET OF LINEAR
      C      EQUATIONS WITH AUGMENTED MATRIX OF COEFFICIENTS IN THE N BY
      C      N+1 ARRAY A AND IN ADDITION COMPUTES THE INVERSE OF THE
      C      COEFFICIENT MATRIX IN PLACE AS ABOVE. IF INDIC IS POSITIVE,
      C      THE SET OF LINEAR EQUATIONS IS SOLVED BUT THE INVERSE IS NOT
      C      COMPUTED IN PLACE. THE GAUSS-JORDAN COMPLETE ELIMINATION METHOD
      C      IS EMPLOYED WITH THE MAXIMUM PIVOT STRATEGY. ROW AND COLUMN
      C      SUBSCRIPTS OF SUCCESSIVE PIVOT ELEMENTS ARE SAVED IN ORDER IN
      C      THE IROW AND JCOL ARRAYS RESPECTIVELY. K IS THE PIVOT COUNTER,
      C      PIVOT THE ALGEBRAIC VALUE OF THE PIVOT ELEMENT, MAX
      C      THE NUMBER OF COLUMNS IN A AND DETER THE DETERMINANT OF THE
      C      COEFFICIENT MATRIX. THE SOLUTIONS ARE COMPUTED IN THE (N+1)TH
      C      COLUMN OF A AND THEN UNSCRAMBLED AND PUT IN PROPER ORDER IN
      C      X(1)...X(N) USING THE PIVOT SUBSCRIPT INFORMATION AVAILABLE
      C      IN THE IROW AND JCOL ARRAYS. THE SIGN OF THE DETERMINANT IS
      C      ADJUSTED, IF NECESSARY, BY DETERMINING IF AN ODD OR EVEN NUMBER
      C      OF PAIRWISE INTERCHANGES IS REQUIRED TO PUT THE ELEMENTS OF THE
      C      JORD ARRAY IN ASCENDING SEQUENCE WHERE JORD(IROW(I)) = JCOL(I).
      C      IF THE INVERSE IS REQUIRED, IT IS UNSCRAMBLED IN PLACE USING
      C      Y(1)...Y(N) AS TEMPORARY STORAGE. THE VALUE OF THE DETERMINANT
      C      IS RETURNED AS THE VALUE OF THE FUNCTION. SHOULD THE POTENTIAL
      C      PIVOT OF LARGEST MAGNITUDE BE SMALLER IN MAGNITUDE THAN EPS,
      C      THE MATRIX IS CONSIDERED TO BE SINGULAR AND A TRUE ZERO IS
      C      RETURNED AS THE VALUE OF THE FUNCTION.
      C
      C      REFERENCES CARNAHAN, LUTHER AND WILKES (1969)
      C      APPLIED NUMERICAL METHODS. WILEY, NEW YORK.
      C
      C      FUNCTION SIMUL (N,A,X,EPS,INDIC,NRC)
      C      DIMENSION IROW(50), JCOL(50), JORD(50), Y(50), A(NRC,NRC), X(N)
      C
      C      MAX = N
      C      IF ( INDIC.GE.0 ) MAX = N + 1
      C
      C      .....IS N LARGER THAN 50 .....
      C      IF ( N.LE.50 ) GO TO 5
      C      WRITE( 6,200)
      C      SIMUL = 0.

```

44		RETURN	SI
45	C		SI
46	C	 BEGIN ELIMINATION PROCEDURE	SI
47	5	DETER = 1.	SI
48		DO 18 K = 1, N	SI
49		KM1 = K - 1	SI
50	C		SI
51	C	 SEARCH FOR THE PIVOT ELEMENT	SI
		PIVOT = 0.	SI
		DO 11 I = 1, N	SI
54		DO 11 J = 1, N	SI
55	C	 SCAN IROW AND JCOL ARRAYS FOR INVALID PIVOT SUBSCRIPTS	SI
56		IF (K.EQ.1) GO TO 9	SI
57		DO 8 ISCAN = 1, KM1	SI
		DO 8 JSCAN = 1, KM1	SI
		IF (I.EQ.IROW(ISCAN)) GO TO 11	SI
60	8	IF (J.EQ.JCOL(JSCAN)) GO TO 11	SI
61	9	IF (ABS(A(I,J)).LE. ABS(PIVOT)) GO TO 11	SI
62		PIVOT = A(I,J)	SI
63		IROW(K) = I	SI
64		JCOL(K) = J	SI
65	11	CONTINUE	SI
66	C		SI
67	C	 INSURE THAT SELECTED PIVOT IS LARGER THAN EPS	SI
68		IF (ABS(PIVOT).GT.EPS) GO TO 13	SI
69		SIMUL = 0.	SI
70		RETURN	SI
71	C		SI
72	C	 UPDATE THE DETERMINANT VALUE	SI
73	13	IROWK = IROW(K)	SI
74		JCOLK = JCOL(K)	SI
75		DETER = DETER*PIVOT	SI
76	C		SI
77	C	 NORMALIZE PIVOT ROW ELEMENTS	SI
78		DO 14 J = 1, MAX	SI
79	14	A(IROWK,J) = A(IROWK,J)/PIVOT	SI
80	C		SI
81	C	 CARRY OUT ELIMINATION AND DEVELOP INVERSE	SI
82		A(IROWK,JCOLK) = 1./PIVOT	SI
		DO 18 I = 1, N	SI
		AIJCK = A(I,JCOLK)	SI
		IF (I.EQ.IROWK) GO TO 18	SI
86		A(I,JCOLK) = - AIJCK/PIVOT	SI
87		DO 17 J = 1, MAX	SI
88	17	IF (J.NE.JCOLK) A(I,J) = A(I,J) - AIJCK*A(IROWK,J)	SI
89	18	CONTINUE	SI
90	C		SI
91	C	 ORDER SOLUTION VALUES (IF ANY) AND CREATE JORD ARRAY	SI
92		DO 20 I = 1, N	SI
93		IROWI = IROW(I)	SI
94		JCOLI = JCOL(I)	SI
95		JORD(IROWI) = JCOLI	SI
96	20	IF (INDIC.GE.0) X(JCOLI) = A(IROWI,MAX)	SI
97	C		SI
98	C	 ADJUST SIGN OF DETERMINANT	SI
99		IF(N.EQ.1) GO TO 24	SI
100		INTCH = 0	SI
101		NM1 = N - 1	SI
102		DO 22 I = 1, NM1	SI
103		IP1 = I + 1	SI
104		DO 22 J = IP1, N	SI
105		IF (JORD(J).GE.JORD(I)) GO TO 22	SI
106		JTEMP = JORD(J)	SI

107		JORD(J) = JORD(I)	SI
108		JCRD(I) = JTEMP	SI
		INTCH = INTCH + 1	SI
	22	CONTINUE	SI
111		IF (INTCH/2*2.NE.INTCH) DETER = - DETER	SI
112	C		SI
113	C	 IF INDIC IS POSITIVE RETURN WITH RESULTS	SI
114	24	IF (INDIC.LE.0) GO TO 26	SI
		SIMUL = DETER	SI
		RETURN	SI
117	C		SI
118	C	 IF INDIC IS NEGATIVE OR ZERO, UNSCRAMBLE THE INVERSE	SI
119	C	FIRST BY ROWS	SI
120	26	DO 28 J = 1, N	SI
121		DO 27 I = 1, N	SI
122		IROWI = IROW(I)	SI
123		JCOLI = JCOL(I)	SI
124	27	Y(JCOLI) = A(IROWI,J)	SI
125		DO 28 I = 1, N	SI
126	28	A(I,J) = Y(I)	SI
127	C	 THEN BY COLUMNS	SI
128		DO 30 I = 1, N	SI
129		DO 29 J = 1, N	SI
130		IROWJ = IROW(J)	SI
131		JCOLJ = JCOL(J)	SI
132	29	Y(IROWJ) = A(I,JCOLJ)	SI
133		DO 30 J = 1, N	SI
134	30	A(I,J) = Y(J)	SI
135	C		SI
136	C	 RETURN FOR INDIC NEGATIVE OR ZERO	SI
137		SIMUL = DETER	SI
138		RETURN	SI
139	C		SI
	C	 FORMAT FOR OUTPUT STATEMENT	SI
	200	FORMAT(10HON TOO BIG)	SI
142	C		SI
143		END	SI

PR1,S FCHFILE1.MVFIX1,.VFIX1-S,.ENDS/VFIX1-S,.PRESS/VFIX1-S

337 FCHFILE1(1).VFIX1-S

2	C	THIS PROGRAM COMPUTES PROFILER CORRECTED VERTICAL
3	C	VELOCITY BY USING THE PRESSURE DERIVATIVE
4	C	BEFORE EXECUTING, THE INPUT DATA FEB FILE MUST BE ASSIGNED
5	C	TO SOME UNIT NUMBER, AND AN OUTPUT FILE ASSIGNED. TIME
6	C	INTERPOLATION OF THE INPUT FEB FILE SHOULD HAVE BEEN
7	C	COMPLETED (HTP*PROG,ZINTERP)
8	C	
9	C	MAPPING ELEMENT: FCHFILE1.MVFIX1-S
10	C	
11		DIMENSION WI(500),P(17)
12		COMMON KSEG, IEND, K1, K2, IND, NVAR, NSEG, NSSEG, IUNR, IUN, NUMV
13		COMMON /RHOR/LR, NR, NBR, NMBR, NMFR, NFR, NIR, NAR, IPR(20)
14		COMMON /RDATA/VR(2000)
15		COMMON /RDOCF/FDOCR(50)/RDOCI/IDOCR(50)/RDOCA/ADOCR(200)
16		COMMON /DIAGS/MSGR, MSGW, NNNR, NNNW, NWIP, NNF, NNI, V, TRST, INST
17		DATA NNNR, NNNW, NNIP, NNF, NNI, NNA/5000, 5000, 20, 50, 50, 200/
18	C	
19	C	SET INITIAL DATA (K= NUMBER OF POINTS ON EACH SIDE OF
20	C	THE PRESSURE DATA POINT TO BE USED IN COMPUTING THE
21	C	DERIVATIVE; DELT= TIME INTERVAL BETWEEN PRINTS)

```

22 C
23 K=8
24 K1=K+1
25 K2=2*K+1
26 DELT=.0625
27 RHOG1=.9955
28 COEFF=3./(K*(K+1)*(2*K+1)*DELT)
29 MSGR=0
30 MSGW=0
31 C
32 WRITE(6,2)
33 2 FORMAT(' INPUT UNIT, OUTPUT UNIT, NUMBER SEGMENT S, START SEGMENT,
34 3, 4, 5 PRESSURE VARIABLE NUMBER, VERTICAL VELOCITY VARIABLE NUMBER')
35 READ(5,1)IUNR,IUNW,NSEG,NSSEG,NVAR,NUMV
36 1 FORMAT(1)
37 KFLAG=0
38 ISTART=K+1
39 CALL ZREAD(IUNR,IF,NSSEG)
40 IEND=NR
41 DO 30 NSEG=1,NSSEG
42 C
43 C FOR FIRST SEGMENT, CALL SUBROUTINE TO SET FIRST K DERIVATIVES TO ZERO.
44 C FOR LAST SEGMENT, CALL SUBROUTINE TO SET LAST K DERIVATIVES TO ZERO.
45 C
46 IF(KSEG.EQ.1)CALL ENDS(WI)
47 IF(KSEG.NE.1)ISTART=1
48 IF(KSEG.EQ.NSEG)CALL ENDS(WI)
49 C
50 DO 10 J=ISTART,IEND
51 C
52 C CALL SUBROUTINE TO READ SPAN OF PRESSURE VALUES INTO ARRAY P
53 C
54 CALL PRESS(P,KFLAG)
55 KFLAG=1
56 SUM=0.
57 C
58 C COMPUTE PRESSURE DERIVATIVE
59 C
60 DO 20 II=-K,K
61 20 SUM=SUM+II*P(K1+II)
62 DPOT=COEFF*SUM
63 WI(J)=RHOG1*DPOT
64 C
65 10 CONTINUE
66 C
67 C CALL SUBROUTINE TO WRITE NEW FEB FILE
68 C
69 IF (KSEG.EQ.NSEG)GO TO 25
70 NSSEG=NSSEG+1
71 CALL ZREAD(IUNR,IF,NSSEG)
72 25 CALL WRITE(WI)
73 WRITE(6,3)NSSEG
74 3 FORMAT(' COMPLETED SEGMENT #',I6)
75 IF(KSEG.EQ.NSEG)GO TO 30
76 NSSEG=NSSEG+1
77 CALL ZREAD(IUNR,IF,NSSEG)
78 30 CONTINUE
79 END

```

137 **FCHFILE(1) ENDS/VFIX1-S

```

2 C THIS SUBROUTINE SETS THE FIRST K PRESSURE DERIVATIVES TO ZERO FOR
C THE FIRST SEGMENT, AND THE LAST K EQUAL TO ZERO FOR THE LAST

```

```

3      C      SEGMENT
4      C
5      SUBROUTINE ENDS(WI)
6      DIMENSION WI(1)
7      COMMON KSEG, IEND, K, K2, IND, NVAR, NSEG, NSSEG, IUNR, IUNW, NUMV
8      COMMON /RHDR/LR, NR, NBR, NMFR, NFR, NIR, NAR, IPR(20)
9      COMMON /RDATA/VR(20000)
10     COMMON /RDOCF/FDOCR(50)/RDOCI/IDOCR(50)/RDOCA/ADOCR(200)
11     COMMON /DIAGS/MSGR, MSGW, NNNR, NNNW, NNIP, NNF, NNI, NNA, IRST, IWST
12     C
13     ISTOP=NR-K+1
14     II=ISTOP
15     III=NR
16     C
17     SET LAST CYCLE NUMBER FOR LAST SEGMENT
18     C
19     IF(KSEG.EQ.NSEG) IEND=ISTOP-1
20     C
21     IF(KSEG.EQ.1) II=1
22     IF(KSEG.EQ.1) III=K
23     DO 10 I=II, III
24     10     WI(I)=0.
25     RETURN
26     END

```

E33 *FCHFILE(11).PRESS/VFIX1-S

```

3      C      THIS SUBROUTINE READS A SPAN OF PRESSURE INTO ARRAY P
4      C
5      SUBROUTINE PRESS(P, KFLAG)
6      DIMENSION P(1)
7      COMMON KSEG, IEND, K, K2, IND, NVAR, NSEG, NSSEG, IUNR, IUNW, NUMV
8      COMMON /RHDR/LR, NR, NBR, NMFR, NFR, NIR, NAR, IPR(20)
9      COMMON /RDATA/VR(20000)
10     COMMON /RDOCF/FDOCR(50)/RDOCI/IDOCR(50)/RDOCA/ADOCR(200)
11     COMMON /DIAGS/MSGR, MSGW, NNNR, NNNW, NNIP, NNF, NNI, NNA, IRST, IWST
12     C
13     IF(KFLAG.EQ.1) GO TO 11
14     C
15     FOR FIRST CALL, READ IN 2*K+1 PRESSURE VALUES
16     C
17     DO 10 M=1, K2
18     10     IND=(M-1)*LR+NVAR
19     P(M)=VR(IND)
20     CONTINUE
21     RETURN
22     C
23     FOR CALLS OTHER THAN THE FIRST, SHIFT PRESSURE VALUES DOWN BY ONE
24     C
25     11     K1=K2-1
26     DO 20 M=1, K1
27     20     P(M)=P(M+1)
28     C
29     IND=IND+LR
30     ICHK=LR*NBR
31     C
32     IF AT END OF SEGMENT, CALL SUBROUTINE TO READ IN NEXT SEGMENT
33     AND READ IN LAST PRESSURE VALUE IN SPAN
34     C
35     IF(IND.GT.ICHK) CALL NEXSEG
36     P(K2)=VR(IND)
37     C
38     RETURN
39     END

```

7,5 FCHFILE1,NEXSEG/VFIX1-S,,WRITE/VFIX1-S,,MVFIX1-S

E33*FCHFILE1(1).NEXSEG/VFIX1-S

```

C      THIS SUBROUTINE READS THE NEXT SEGMENT AND RESETS THE INDEX
C
C      SUBROUTINE NEXSEG
C      COMMON KSEG,IEND,K,K2,IND,NVAR,NSEG,NSSEG,IUNR,IUNW
C      COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(20)
C      COMMON /RDATA/VR(20000)
C      COMMON /RDOCF/FDOCR(50)/RDOCI/IDOCR(50)/RDOCA/ADOCR(200)
C      COMMON /DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
C
C      NSSEG=NSSEG+1
C      CALL ZREAD(IUNR,IF,NSSEG)
C      IND=NVAR
C      RETURN
C      END

```

E33*FCHFILE1(1).WRITE/VFIX1-S

```

C      THIS SUBROUTINE WRITES THE NEW FEB FILE
C      (THE OUTPUT FILE CONTAINS 2 MORE VARIABLES THAN THE INPUT FILE)
C
C      SUBROUTINE WRITE(WI)
C      DIMENSION WI(1)
C      COMMON KSEG,IEND,K,K2,IND,NVAR,NSEG,NSSEG,IUNR,IUNW,NUMV
C      COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(20)
C      COMMON /RDATA/VR(20000)
C      COMMON /RDOCF/FDOCR(50)/RDOCI/IDOCR(50)/RDOCA/ADOCR(200)
C      COMMON /DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
C      COMMON /WHDR/LW,NW,NBW,NMBW,NMFW,NFW,NIW,NAW,IPW(20)
C      COMMON /WDATA/VW(20000)
C      COMMON /WDOCF/FDOCW(50)/WDOCI/IDOCW(50)/WDOCA/ADOCW(200)
C
C      DO 10 I=1,NNF
C      10      FDOCW(I)=FDOCR(I)
C      DO 20 I=1,NNI
C      20      IDOCW(I)=IDOCR(I)
C      DO 30 I=1,NNA
C      30      ADOCW(I)=ADOCR(I)
C      NW=NR
C      NBW=NBR
C      NMBW=NMBR
C      NMFW=NMFR
C      NFW=NFR
C      NIW=NIR
C      NAW=NAR
C      LW=LR+2
C
C      VARIABLES BEFORE VERTICAL VELOCITY ARE WRITTEN OUT THE SAME
C
C      NUMV1=NUMV-1
C      DO 40 J=1,NUMV1
C      40      IPW(J)=IPR(J)
C      DO 40 I=1,NR
C      40      IND1=(I-1)*LR
C      40      IND2=(I-1)*LW
C      40      VW(IND2+J)=VR(IND1+J)
C
C      VARIABLE W=V-WI IS WRITTEN OUT JUST BEFORE THE VERTICAL VELOCITY.
C      VARIABLE WI IS WRITTEN OUT AT THE END OF THE INPUT VARIABLES.
C
C      DO 50 I=1,NR
C      50      IND1=(I-1)*LR

```

```

45      IND2=(I-1)*LV
46      VW(IND2+LV)=VI(I)*100
47      50      VW(IND2+NUMV)=VR(IND1+NUMV)-VI(I)*100
48      C
49      NUMV=NUMV+1
50      LV=LV-1
51      C
52      C      VARIABLES FROM VERTICAL VELOCITY THROUGH THE LAST INPUT VARIABLE
53      C      ARE WRITTEN OUT SHIFTED OVER ONE LOCATION
54      DO 60 J=NUMV, LV
55          IPW(J)=IPR(J-1)
56      DO 60 I=1, NR
57          IND1=(I-1)*LR
58          IND2=(I-1)*LV
59          VW(IND2+J)=VR(IND1+J-1)
60      C
61      IPW(NUMV)=*W
62      IPW(LV)=*VI
63      CALL ZWRIT(IUNW, IF, 0)
64      RETURN
65      END

```

3A=LIB(1).UNORTHOG

```

1      C
2      C*****
3      C PROGRAM : UNORTHOG
4      C PURPOSE : TO CONVERT PROFILE FILES WITH ORTHOGONAL
5      C            INSTRUMENT VELOCITIES TO FILES WITH THE
6      C            ORIGINAL VELOCITIES IN ACOUSTIC AXES COORDS.
7      C*****
8      C
9      C
10     COMMON/BEIA/ORTHOG(4,3),V(3)
11     C COMMONS FOR ZREAD
12     COMMON/KHDR/LR,NR,NFR,NHR,NHNR,NHFR,NFR,NIR,NAR,IPR(15)
13     COMMON/KDATA/VP(15,500)
14     COMMON/RDOCF/DOCCR(40)
15     COMMON/RDOC1/IDOCR(20)
16     COMMON/RDOCA/ADOCR(50)
17     C
18     COMMON/DIAGS/MSGR,MSGW,NNNR,NNNV,NNIP,NNF,NNI,NNA,IRST,IMST
19     C
20     C INITIALIZE CONTROL HEADER
21     DATA LR,NR,NFR,NIR,NAR/15,500,40,20,50/
22     DATA NNAR,NNIP,NFR,NNI,NNA/500,15,40,20,50/
23     C INITIALIZE VELOCITY ORTHOGONALIZATION MATRIX
24     DATA ORTHOG(1,1),ORTHOG(2,1),ORTHOG(3,1),ORTHOG(4,1)/
25     $ .534,.02485,-.67493,.65760/
26     DATA ORTHOG(1,2),ORTHOG(2,2),ORTHOG(3,2),ORTHOG(4,2)/
27     $ .3505,.66162,.01352,.63666/
28     DATA ORTHOG(1,3),ORTHOG(2,3),ORTHOG(3,3),ORTHOG(4,3)/
29     $ .2268,.03198,.70948,.60185/
30     C
31     MSGR=1
32     MSGW=1
33     C*****
34     C ENTER INPUT AND OUTPUT FILE INFORMATION
35     C*****
36     C
37     WRITE(6,200)
38     200 FORMAT(// ' ENTER: NUIN1,NSEG1,NSSEG1' )
39     READ(5,300) NUIN1,NSEG1,NSSEG1
40     300 FORMAT(//

```

```

41      WRITE(6,400) NUIN1,NSSEG1,NSSEG1
42      400  FORMAT(' NUIN1 = ',I2,' NSEG1 = ',I5,' NSSEG1 = ',I5)
43      700  FORMAT(' NUIN1 = ',I2,' NSEG2 = ',I5,' NSSEG2 = ',I5,
44      *          ' MSG12 = ',I2)
45      C
46      C READ SPECIFICATIONS FOR OUTPUT FILE
47      C
48      WRITE(6,500)
49      500  FORMAT(' ENTER: NUOUT ')
50      READ(5,300) NUOUT
51      WRITE(6,5000) NUOUT
52      5000 FORMAT(' NUOUT = ',I2)
53      C
54      C*****
55      C* BEGIN MAIN SECTION OF PROGRAM HERE
56      C*****
57      C
58      C
59      DO 1000 LOOP = NSSEG1, NSSEG1
60          CALL ZREAD(NUIN1,IF,LOOP)
61          DO 2000 I = 1, NR
62              C
63              C*****
64              C* CONVERT VELOCITIES BACK TO ORIGINAL ACOUSTIC COORDS
65              C*****
66              C
67              DO 2500 K = 1, 3
68                  W(K) = VR(K+4,I)
69                  IF(I.LT.3) WRITE(6,7001) K,W(K)
70                  7001  FORMAT(' K = ',I3,' W(K) = ',G10.6)
71                  2500  CONTINUE
72                  DO 2600 K = 1, 3
73                      V = -ORTHOG(1,K)
74                      DO 2601 J = 1, 3
75                          V = V+W(J)*ORTHOG(J+1,K)
76                  2601  CONTINUE
77                      VR(K+4,I) = V
78                  C
79                  IF(I.LT.3) WRITE(6,7002) I,K,V
80                  7002  FORMAT(' I = ',I2,' K = ',I2,' V(K) = ',G10.6)
81                  2600  CONTINUE
82                  CALL ZWRIT(-NUOUT,IF,3)
83                  1000  CONTINUE
84                  STOP
85                  END

```

BPRT,S V.PREFIX

2A*LIB(1).TSEXPLOT2

```

1      C PROGRAM TSEXPLOT2
2      C
3      C JIM VEGA CSC SEPT. 1981
4      C
5      C THIS PROGRAM PRODUCES PLOTS OF ANY FEB VARIABLE
6      C VS EITHER TIME OR CYCLE NUMBER.
7      C
8      C*****
9      C THIS PROGRAM IS EXECUTED BY THE FOLLOWING:
10     C
11     C 2XQT VEGA*LIB.TSEXPLOT2
12     C
13     C FOLLOWED BY TWO DATA CARDS FOR EACH SUBPLOT.

```

```

14 C DATA CARD 1 CONTAINS:
15 C
16 C IU, IABSYS, IP, YMAX, YSTP, YMIN, IDEC
17 C
18 C WHERE:
19 C IU - LOGICAL UNIT NUMBER OF INPUT FILE.
20 C IABSYS - POINTER DETERMINING ABSCISSA VALUES
21 C 0 - CYCLE NUMBER
22 C 1 - RELATIVE TIME (IN SECONDS)
23 C IP - POSITION OF ORDINATE VARIABLE IN FEB
24 C DATA ARRAY VR.
25 C YMAX - MAXIMUM EXPECTED VALUE OF ORDINATE VARIABLE
26 C YSTP - ORDINATE LABELING INTERVAL
27 C YMIN - MINIMUM EXPECTED VALUE OF ORDINATE VARIABLE
28 C (VALUE OF ORDINATE AT ORIGIN)
29 C IDEC - DECIMATION
30 C
31 C * - THESE VARIABLES ARE FLOATING POINT RATHER THAN INTEGER
32 C
33 C *** FOR A PLOT AGAINST CYCLE NUMBER (IABSYS=0) ***
34 C DATA CARD 2 CONTAINS:
35 C
36 C NUMSEG, IBEGIN, CYCIN
37 C
38 C WHERE:
39 C NUMSEG - NUMBER OF SEGMENTS TO BE PLOTTED
40 C IBEGIN - SEGMENT NUMBER OF FIRST SEGMENT TO BE PLOTTED
41 C CYCIN - NUMBER OF CYCLES PER INCH OF PLOT
42 C
43 C NUMSEG AND IBEGIN ARE INTEGERS, CYCIN IS FLOATING POINT
44 C
45 C *** FOR A PLOT AGAINST RELATIVE TIME (IABSYS=1) ***
46 C DATA CARD 2 CONTAINS:
47 C
48 C ISTSEC, INDESEC, IPTIME
49 C
50 C WHERE:
51 C ISTSEC - PLOT START TIME (IN RELATIVE SECONDS)
52 C INDESEC - PLOT END TIME (IN RELATIVE SECONDS)
53 C IPTIME - LOCATION OF RELATIVE TIME VARIABLE IN
54 C THE FEB DATA ARRAY VR
55 C
56 C ALL THREE PARAMETERS ARE INTEGERS.
57 C
58 C THE FOLLOWING DATA CARD SHOULD FOLLOW THE SECOND
59 C DATA CARD OF THE LAST SUBPLOT:
60 C
61 C 99, 0, 0, 0, 0, 0, 0
62 C
63 C TO INSURE PROPER TERMINATION OF BOTH THE PROGRAM
64 C AND THE PLOT,
65 C
66 C ***** NOTES *****
67 C
68 C THIS PROGRAM IS SET UP TO USE THE 34 INCH PAPER
69 C ON THE ZETA PLOTTER, AS A RESULT THE MAXIMUM NUMBER
70 C OF SUBPLOTS PER PLOT IS 5.
71 C
72 C THIS PROGRAM PRODUCES AN INTERMEDIATE PLOT FILE ON
73 C UNIT 25. IT IS NECESSARY TO EXECUTE ONE OF THE DISPLA
74 C POST-PROCESSORS TO OBTAIN A PLOT TAPE.
75 C
76 C IT IS HIGHLY RECOMMENDED THAT THE INTERMEDIATE PLOT FILE

```

```

77 C ON UNIT 25 BE COPIED TO ANOTHER FILE IMMEDIATELY FOLLOWING
78 C EACH EXECUTION TO GUARD AGAINST LOSS.
79 C
80 C
81 C *****
82 C WRITE (6,1)
83 1 FORMAT ('FOR INSTRUCTIONS--TERMINATE--RUN--AND--ADD',
84 S ' VEGA+LIB,INST/PLOT2')
85 C
86 C COMMONS FOR ZREAD
87 COMMON/RHDR/LR,NR,NFR,NBR,NMFR,NMR,NIR,NAR,IPR(15)
88 COMMON/RDCCF/ROCCP(50)/ROCCI/IDOCR(50)/ROCCA/ADOCR(100)
89 COMMON/RDATA/VR(15,500)
90 COMMON/DIAGS/MSGP,MSGW,NNNR,NNNW,INIP,NMF,NNI,4NA,IRST,IWST
91 C
92 COMMON/PLOT/IU,IP,YMAX,YSTP,YMIN,IDEC,YORIG
93 C
94 C INITIALIZE FEB READ
95 DATA LR,NR,NFR,NIR,NAR/20,500,50,50,100/
96 DATA NNR,NNIP,NMF,NNI,NNA/500,20,50,57,100/
97 YORIG=1.00
98 C
99 C SET UP PLOT PAGE
100 CALL COMPR
101 CALL BGNPL (0)
102 CALL SCMLX
103 CALL NOEPDR
104 CALL FLATBD
105 CALL PAGE (72.0,34.0)
106 CALL TITLE (0,0,0,0,0,0,1.0,1.0)
107 CALL GRAF (0,0,1.0,1.0,0.0,1.0,1.0)
108 CALL ENDGR (0)
109 C
110 20 CONTINUE
111 C READ INPUT PARAMETERS
112 READ (5,1000) IU,IABSYS,IP,YMAX,YSTP,YMIN,IDEC
113 IF (IU.EQ.99) GO TO 998
114 C
115 IF (IABSYS.EQ.1) GO TO 50
116 C
117 READ (5,1000) NUMSEG,IREGIN,CYCIN
118 XAXIS=NUMSEG/4CYCIN/500.1
119 IF (XAXIS.GT.71.) GO TO 900
120 GO TO 60
121 C
122 50 CONTINUE
123 READ (5,1000) ISTSEC,INDSEC,IPTIME
124 C
125 60 CONTINUE
126 C
127 IF (IABSYS.NE.1) CALL CYCLNM (NUMSEG,IREGIN,CYCIN,XAXIS)
128 IF (IABSYS.EQ.1) CALL FELSEC (ISTSEC,INDSEC,IPTIME)
129 YORIG=YORIG+6.75
130 GO TO 20
131 C
132 900 WRITE (6,901) XAXIS
133 GO TO 20
134 901 FORMAT ('XAXIS--TOO LARGE--XAXIS=',F7.2,/,
135 S ' MAXIMUM ALLOWABLE SIZE IS 71 INCHES',/,
136 S ' SURPLOT WILL BE OMITTED')
137 C
138 998 CONTINUE
139 C

```

```

140 -----WRITE(6,777)-----
141       777   FORMAT(' CALL DONEPL')
142 -----CALL DONEPL-----
143       1000  FORMAT ( )
144       C
145 -----END-----

```

BPRT,S V.CYCLNM/PLOT2

BA+LIB(1).CYCLNM/PLOT2

```

1      SUBROUTINE CYCLNM (NUMSEG,IBEGIN,CYCIN,XAXIS)
2      C
3      C THIS SUBROUTINE IS CALLED FROM TSERPLOT2 TO
4      C PRODUCE PLOTS OF ANY FCB VARIABLE VS CYCLE NUMBER.
5      C
6      C JIM VEGA      CSC      SEPT. 1981
7      C
8      C *****
9      C
10     C COMMONS FOR ZREAD
11     COMMON/RMDR/LR,NP,NBR,NMR,NFR,NIR,NAR,IPR(15)
12     COMMON/RDOCF/FDOCR(40)/RDOCI/IDOCR(20)/RDOCA/ADOCR(50)
13     COMMON/RDATA/VR(15,500)
14     COMMON/DIAGS/MSGP,MSGW,NNNR,NNNW,NVIP,NMF,NMI,NNA,IRST,IVST
15     C
16     COMMON/PLOT/IU,IP,YMAX,YSTP,YMIN,IDEC,YORIS
17     C
18     DIMENSION XARAY(1000),YARAY(1000)
19     DIMENSION LAB(7),LBLX(3)
20     C
21     C
22     INDX=1
23     NCYCLE=1
24     IB=IBEGIN
25     IFIRST=1
26     IPLT=0
27     LSTSEG=NUMSEG+IBEGIN-1
28     XDEC=IDEC+1,C
29     C
30     20 CONTINUE
31     IF (IFIRST.EQ.1) GO TO 25
32     IF (NCYCLE.LT.NR) GO TO 30
33     25 CALL ZREAD (IU,IF,IB)
34     NCYCLE=1
35     IF (IFIRST.EQ.1.AND.IF.NE.0) GO TO 998
36     IF (ITERM.EQ.1) GO TO 100
37     IF (IF.NE.0) GO TO 100
38     IB=IB+1
39     C
40     30 CONTINUE
41     IF (IFIRST.NE.1) GO TO 50
42     C INITIALIZE SUBPLOT ON FIRST PASS
43     INCDE (13,3000,LBLX)
44     3000 FORMAT ('CYCLE NUMBER')
45     CALL XINTAX
46     CALL LABEL (IPR(IP),NMR,FDOCR(10),IDOCR(11),IP,LAB,LBLX)
47     CALL PHYSOR (1,25,YORIS)
48     CALL TITLE (LAB,100,LBLX,100,LELY,6,XAXIS,4.5)
49     XMIN=IBEGIN+500,-999
50     XSTP=CYCIN+2,0
51     YMAX=NUMSEG+500,+XMIN
52     CALL GRAF (XMIN,XSTP,XMAX,YMIN,YSTP,YMAX)

```

```

53      C
54      C   FRAME SUBPLOT
55          XLNGTH=XAXIS*2.0
56          CALL STRIPT (-1.00,-.75)
57          CALL CONNPT (-1.0,-.75)
58          CALL CONNPT (XLNGTH,.25)
59          CALL CONNPT (XLNGTH,-.75)
60          CALL CONNPT (-1.00,-.75)
61      C
62      C
63          IFIRST=0
64      C
65      50  CONTINUE
66      C   SET UP AXIS ARRAYS
67          XARAY(INDX)=XDEC+INDX*(XDEC-1.)*IPLT/500.
68          YARAY(INDX)=YR(IP,NCYCLE)
69      C
70          INDX=INDX+1
71          NCYCLE=NCYCLE+IDEC
72      C
73          IF (IDOCR(1).EQ.LSTSEG) ITERM=1
74          IF (IDOCR(1).EQ.1) ITERM=1
75      C
76          IF (INDX.LT.1001) GO TO 20
77      C
78          INDX=INDX-1
79          CALL CURVE (XARAY,YARAY,INDX,0)
80          INDX=1
81          IPLT=IPLT+1
82          IF (ITERM.NE.1) GO TO 20
83      C
84      100  CONTINUE
85      C   TERMINATE SUBPLOT
86          CALL CURVE (XARAY,YARAY,INDX,0)
87          CALL ENDGR (0)
88          GO TO 1000
89      999  WRITE (6,999) IF
90      999  FORMAT (' ERROR IN ZREAD, IF=',I4)
91      1000  CONTINUE
92          RETURN
93          END

```

BPRT,S V.RELSEC/PLOT2

```

6A+LIB(1).RELSEC/FLOT2
1      SUBROUTINE RELSEC (ISTSEC,INDSEC,IPTIME)
2      C
3      C   THIS SUBROUTINE IS CALLED TO PLOT ANY FER
4      C   VARIABLE AGAINST RELATIVE TIME (RELSEC).
5      C
6      C
7      C   JIM VEGA      CSC      SEPT. 1981
8      C
9      C   *****
10     C
11     C   COMMONS FOR ZREAD
12     COMMON/RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR, (IR,IPI,15)
13     COMMON/RDOCF/FDOCF(50)/FDOCI/IDOCR(50)/RDOC AC CR(100)
14     COMMON/RDATA/VR(15,500)
15     COMMON/DIAGS/MSGR,MSGW,NNNR,NNNV,MNIP,MNF,MMT,VNA,IRST,IMST
16     C
17     COMMON/PLOT/IU,IP,YMAX,YSTP,YMIN,IL,C,YORIG

```

```

18 C
19 DIMENSION XAPAY(1000),YARAY(1000)
20 DIMENSION LAR(7),LBLX(3)
21 C WRITE (6,1114) IU,IP,YMAX,YSTP,YMIN,IDEC
22 C1114 FORMAT (2I5,2X,3F7.2,3X,16)
23 C
24 IB=0
25 ITERM=0
26 NCYCLE=0
27 ICOUNT=0
28 INDY=1
29 IFIRST=1
30 STRTSC=ISTSEC*1.
31 96 WRITE(6,99)
32 99 FORMAT(' NUMBER OF SECONDS PER INCH?')
33 READ(5,98)TINC
34 98 FORMAT(I)
35 XLNGTH=(INDSEC-ISTSEC)/TINC*1.0
36 IF(XLNGTH.GT.71.)WRITE(6,97)
37 97 FORMAT(' X AXIS TOO LONG (MAX=71): REENTER')
38 IF(XLNGTH.GT.71.)GO TO 96
39 C
40 20 CONTINUE
41 IB=IB+1
42 CALL ZREAD (IU,IF,IB)
43 IF (IF.NE.0) GO TO 998
44 IF (IFIRST.EQ.0) GO TO 25
45 C INITIALIZE SUBPLOT ON FIRST PASS
46 ENCODE (7,3000,LBLX) IP(15)
47 3000 FORMAT (A6,'S')
48 CALL XINTAX
49 CALL LABEL (IPR(IP),NMNR,FDOCR(10),IDOCR(11),IP,LAR,LBLY)
50 CALL PHYSOR (1.25,YORIG)
51 C WRITE (6,1113) YORIG
52 C1113 FORMAT (F7.3)
53 CALL TITLE (LAR,100,LBLX,100,LBLY,6,XLNGTH,4,5)
54 CALL YGRAF (STRTSC,TINC,YMIN,YSTP,YMAX)
55 IFIRST=0
56 PRI=NR-IDEC
57 C
58 25 CONTINUE
59 IF (.NOT.(ISTSEC.GE.VR(IPTIME,1).AND.ISTSEC.LE.VR(IPTIME,NR)))
60 S GO TO 20
61 C
62 30 CONTINUE
63 NCYCLE=NCYCLE+1
64 C
65 IF (ISTSEC.GE.VR(IPTIME,NCYCLE)) GO TO 30
66 C
67 50 CONTINUE
68 DELTAT=VR(IPTIME,NCYCLE+IDEC)-VR(IPTIME,NCYCLE)
69 IF (DELTAT.GE.5.C) GO TO 200
70 IF (VR(IPTIME,NCYCLE).GE.INDSEC) GO TO 100
71 C
72 C SET UP AXIS ARRAYS
73 XARAY(INDY)=VR(IPTIME,NCYCLE)
74 YARAY(INDY)=VR(IP,NCYCLE)
75 C WRITE (6,1111)VR(IPTIME,NCYCLE),VR(IP,NCYCLE)
76 C S XARAY(INDY),YARAY(INDY)
77 C1111 FORMAT (4(F7.2,3X))
78 C
79 C WRITE(6,98)IPTIME,NCYCLE,IP
80 IF (.NOT.(NCYCLE.GE.NR)) GO TO 60

```

```

81      IB=IB+1
82      CALL ZREAD (IU,IF,IB)
83      IF (IF.NE.0) GO TO 100
84      ACYCLE=1-IDEC
85      C
86      60      CONTINUE
87      IF (INDX.EQ.1000) GO TO 260
88      C
89      70      CONTINUE
90      NCYCLE=NCYCLE+IDFC
91      INDX=INDX+1
92      GO TO 50
93      C
94      C *****
95      C
96      C PLOT DATA
97      C
98      100     CONTINUE
99      ITERM=1
100     C
101     200     CONTINUE
102     IF (INDX.NE.1000) INDX=INDX-1
103     WRITE (6,1112) ITERM,INDX
104     1112    FORMAT (' CALLING CURVE ITERM=',I3,' INDX=',I6)
105     CALL CURVE (XARAY,YARAY,INDX,0)
106     ICOUNT=ICOUNT+INDX
107     INDX=0
108     C
109     IF (ITERM.NE.1) GO TO 70
110     YFRAME=XLENGTH+2.5
111     C
112     C FRAME SUBPLOT
113     CALL STRPT (-1.00,-.75)
114     CALL CONNPT (-1.0,5.25)
115     CALL CONNPT (XFRAME,5.25)
116     CALL CONNPT (XFRAME,-.75)
117     CALL CONNPT (-1.00,-.75)
118     C
119     CALL ENDGR (0)
120     GO TO 1000
121     C
122     996     WRITE (6,999) IF
123     999     FORMAT (' ERROR IN ZREAD, IF=',I4)
124     C
125     1000    CONTINUE
126     RETURN
127     END

```

BPRT,S.V.LABEL/PLOT2

JAOLIB(1).LABEL/PLOT2

```

1      SUBROUTINE LABEL (NAMVAR,NHBR,DAY,IYR,IP,LAB,L3LY)
2      C
3      C THIS SUBROUTINE ENCODES THE DATA LITERALS USED FOR
4      C TITLES ON THE PLOTS PRODUCED BY TSEBLOT2.
5      C
6      C JIM VEGA      CSC      SEPT. 1981
7      C
8      C *****
9      C
10     C SUBPLOT TITLE
11     C ENCODE (40,3000,LAB) NAMVAR,NHBR,DAY,IYR

```

```

12 C
13 C ORDINATE TITLE
14 IF (IP.EQ.3.OR.IP.EQ.14)-ENCODE(4,3010,L6LY)-NAMVAR
15 IF (IP.EQ.1.OR.IP.EQ.2.OR.IP.EQ.4.OR.IP.EQ.8.OR.
16 IP.EQ.9.OR.IP.EQ.10)-ENCODE(4,3011,L6LY)-NAMVAR
17 IF (IP.EQ.5.OR.IP.EQ.6.OR.IP.EQ.7.OR.IP.EQ.11.OR.
18 IP.EQ.12.OR.IP.EQ.13.OR.IP.EQ.15)-ENCODE(4,3012,L6LY)-NAMVAR
19 C
20 RETURN
21 3000 FORMAT (A6,' CAST ',A3,' ',F8.3,' ',I4,'S')
22 3010 FORMAT (A6,' ')
23 3011 FORMAT (A5,' ')
24 3012 FORMAT (A6)
25 END

```

APRT.S V.MAPPL0T2

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NORDA Technical Note 155	2. GOVT ACCESSION NO. ADA121 498	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Documentation of Software for the Neil Brown Instrument Systems/NORDA Velocity/CTD Profiler		5. TYPE OF REPORT & PERIOD COVERED Final
7. AUTHOR(s) K.D. Saunders Henry Perkins		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Ocean Research and Development Activity NSTL Station, Mississippi 39529		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Ocean Research and Development Activity NSTL Station, Mississippi 39529		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE July 1982
		13. NUMBER OF PAGES 61
		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Distribution Unlimited DISTRIBUTION STATEMENT A Approved for public release; Distribution Unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) oceanographic data processing UNIVAC 1108 profiler velocity CTD		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Software developed for processing Neil Brown Instrument Systems/NORDA profiler data is documented in this report. The software includes programs for translating the profiler data from original NBIS format to engineering units in UNIVAC/NAVOCEANO FEB files and for editing and correcting the data subsequently. This report provides complete descriptions of the programs as well as operating information.		

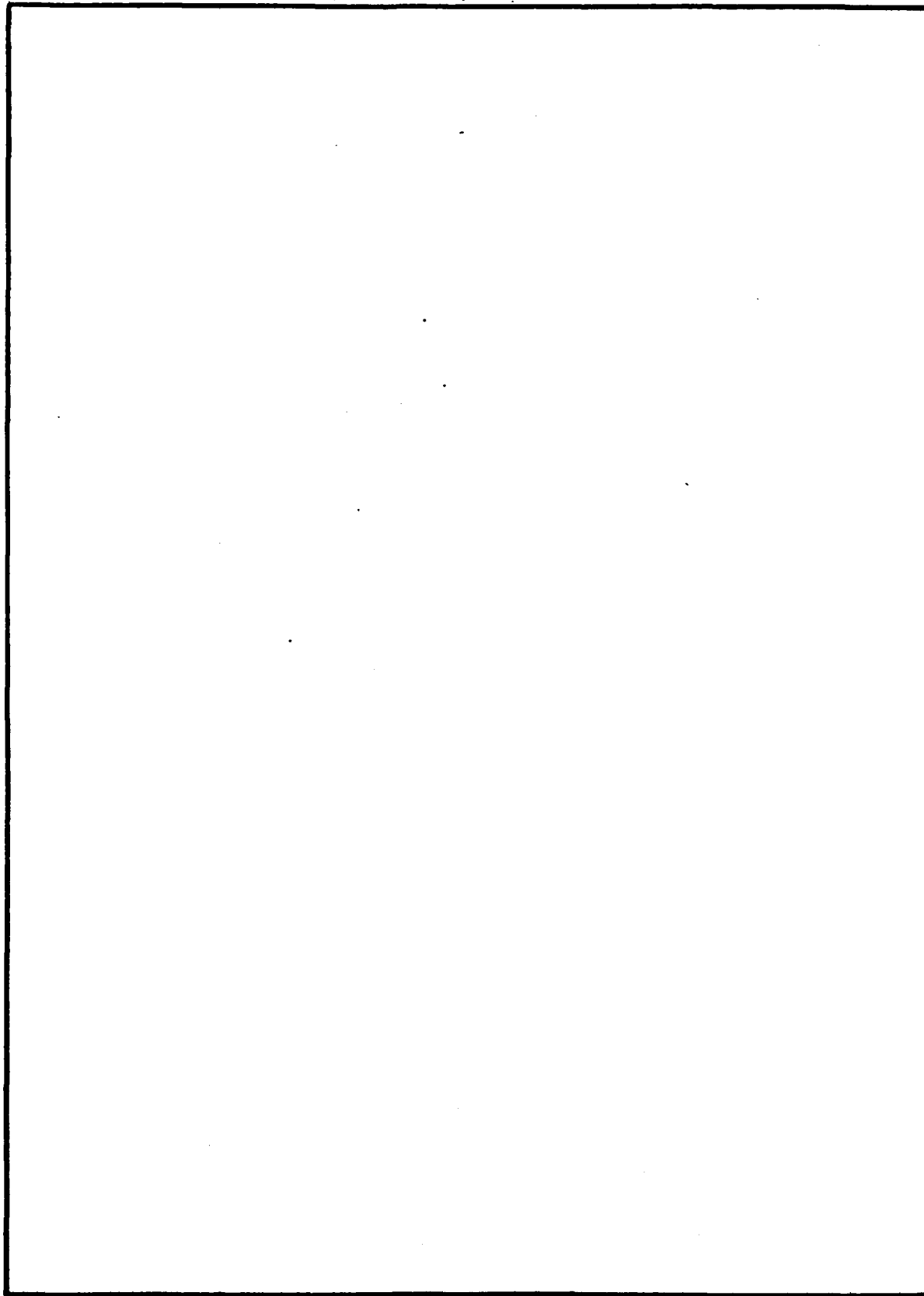
DD FORM 1473 1 JAN 73

EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LP-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

1
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)



SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)