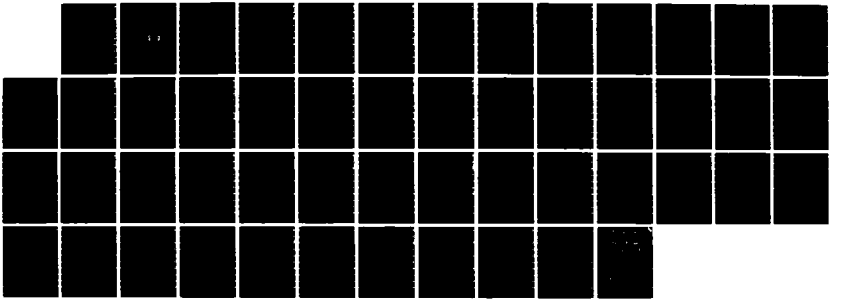
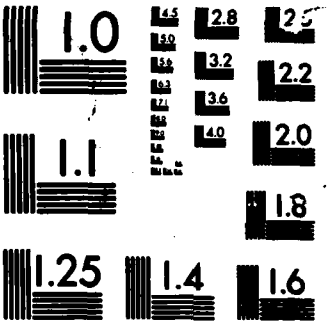
AD-A167 883 A FORTRAN 77 COMPUTER PROGRAM FOR PROCESSING RAW CTD (CONDUCTIVITY TEMPER..(U) NAVAL OCEAN RESEARCH AND DEVELOPMENT ACTIVITY NSTL STATION MS.. W J TEAGUE 1/1

UNCLASSIFIED FEB 86 NORDA-137 F/G 9/2 NL





MICROCOPY CHART

Naval Ocean Research and Development Activity
February 1986 **Report 137**

A FORTRAN 77 Computer Program for Processing Raw CTD Data to ASCII FEB File Format

William J. Teague
Oceanography Division
Ocean Science Directorate

Foreword

The processing of data must keep pace with improvements in computer technology. When the Naval Oceanographic Office converted its basic file structure to ASCII Fast and Easy Binary format it became advantageous for the Naval Ocean Research and Development Activity to rewrite the CTD processing program to match data formats. This report describes the FORTRAN 77 program to accomplish this conversion.

R. P. Onorati, Captain, USN
Commanding Officer, NORDA

Executive summary

A software package written in FORTRAN 77 has been developed for processing raw conductivity, temperature, and depth (CTD) data to ASCII Fast and Easy Binary (FEB) format. The programs contained within the package are machine independent with the exception of the tape block-read routine. Briefly, the tape header information is decoded and checked against information input by the program user, the raw data are read from the data-logger tape and converted to engineering units, the data are screened for spurious data values and, finally, the data are written to FEB files. Testing was performed on VAX 11/750 and UNIVAC 1180 machines.

Keywords: CTDRAW computer program; HEDFIL computer programs.

Acknowledgments

This work was supported by the Physical Oceanography Branch of the Naval Oceanographic Office under Work Request 50045.

Contents

A FORTRAN 77 computer program for processing raw CTD data to ASCII FEB file format

Introduction

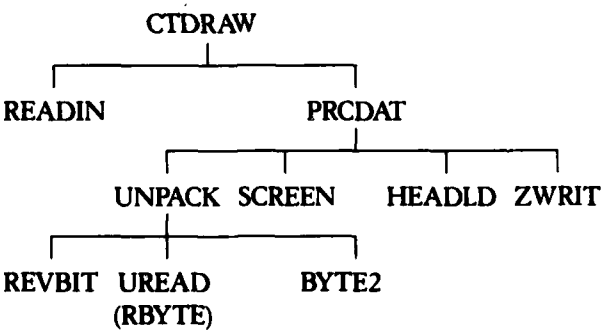
Basic input/output file structure, FEB format (Hallock, 1980), for the physical oceanography branch at the Naval Oceanographic Office (NAVOCEANO) has recently been converted to ASCII FEB format (Teague, 1984). It is now advantageous to rewrite the CTD processing program package in FORTRAN 77. This paper concentrates on the function of the first processing program, RAWRUN, and describes its replacement, CTDRAW. Main reasons for the replacement of RAWRUN by CTDRAW follow.

- RAWRUN is written in FORTRAN-V, a language the NAVOCEANO UNIVAC no longer supports. It is anticipated that by the late 1980s, the ability to compile FORTRAN-V programs on the UNIVAC will be lost, which will ultimately lead to nonworking programs.
- FEB utility programs are in the process of being converted to FORTRAN 77 in order to access ASCII FEB files as opposed to "fieldata" (a six-bit character code native to the UNIVAC) FEB files. New programs will be written in FORTRAN 77.
- RAWRUN is far too complicated and awkward for updating and flexible usage. This program has been patched many times and numerous features have become outdated and unnecessary. It is very machine dependent (UNIVAC), and its present structure cannot be transferred to the NAVOCEANO shipboard computer system so that onboard data processing can parallel inhouse processing.

Discussion

Efficiency, simplicity, and easy use are primary goals in CTDRAW. Input files from the data-logger tapes are read directly. Output FEB file attributes and names are made under program control and files are saved on magnetic tapes under user control. Although run streams (job control) are not program generated (the programs would then be machine dependent), the run streams are simple, allowing for maximum flexibility in the data processing. However, a run-stream-generating program could be written specifically for the UNIVAC for handling the magnetic tape assignments and saving files to tape.

CTDRAW is written entirely in FORTRAN 77 (computer listings are given in Appendix A) and is machine independent with the exception of the tape read subroutines. Testing of CTDRAW was done on the VAX 11/750 and the UNIVAC 1180 machines. The program structure is given below.



The main function of CTDRAW is to read the Digi-Data CTD tapes written for the NBIS MK 3 CTD and to output the data in ASCII FEB file format. No editing of the data is performed in this program; however, the data are screened for those data values that exceed upper or lower bounds and difference tolerances. Diagnostic messages, which display the location within the FEB file of the questionable data values, are printed in a diagnostic log file. These diagnostics are used for editing the data in later processing programs. The main program or driver, CTDRAW, calls the subroutines, which are summarized.

READIN—reads processing and header information from instruction file format (this format is described later). The header information is later compared with the header information available on the internal tape label and diagnostic messages are written where there are disagreements.

PRCDAT—processes the data. Loads the data into the FEB common block.

SCREEN—screens data for values out of range and for values exceeding maximum difference tolerances. Checks time differences between samples against a time tolerance. Prints diagnostics for questionable data values.

HEADLD—loads FEB header commons with the information from the instruction file and tape labels.

ZWRIT—outputs data and headers in FEB file format.

UNPACK—decodes the data and converts the data to engineering units.

RBYTE—reads the data tape and returns 2040 data bytes per tape block (stored one byte per word). Tape blocks are read with VAX machine dependent routines. This routine is for use on the VAX.

UREAD—reads the data tape and returns 2040 data bytes (stored one byte per word). Tape blocks are read with UNIVAC machine-dependent routines. This routine is for use on the UNIVAC 1180.

BYTE2—splits an 8-bit byte into two 4-bit bytes consisting of the left 4 bits and the right 4 bits. It is primarily used for the decoding of the header.

REVBIT—reverses the bits in an 8-bit byte. It is used in the decoding of the time word.

Output file names incorporate the station and/or cruise numbers. For the UNIVAC system for example, station 888004 of Cruise 270485 results in the ASCII FEB file named RCTD270485*888004 and diagnostic log file 888004LOG. For the VAX system, output file names are limited to nine characters; therefore, only the station number is incorporated into the ASCII FEB file name (RCT888004, for example).

Control input to the program is in "instruction file format." In this format, a two-letter key identifies the particular input, which follows this key. Thus, the control input can be ordered by the user and omitted when the default values are appropriate. When conflicts exist between the header information on the internal tape label and that supplied by the instruction file, the instruction file information prevails, and diagnostics are printed. The defaults are such that the program can run without any control input. All stations on the tape are then processed to FEB format. Header information can be completed later with an editor or with a header-fill program (a header-fill program is given in Appendix B). Since the FEB files are direct access, the data need not be rewritten while filling headers. Processing with minimum user input could be particularly useful when processing aboard ship.

All tape control is contained in the subroutines UREAD for the UNIVAC and RBYTE for the VAX. If it is necessary to use other tape handling subroutines, then this routine would require modification. Since this program package is written in FORTRAN 77, it should be easily transferable to other computer systems with FORTRAN 77 compilers.

Program input

Instruction file: a two-letter key (IO, OU, MS, CR, SH, SN, DM, CN, FI, ST, ET, MM, DT, PT, TT, CT, DI, GO, QU) followed by the associated input.

1. $Input/output control
   IO IU
   OU IO
   MS MSGW
2. $Cruise no., ship, sensor serial no., and deployment mode
   CR ICRUIS
   SH SHIP
   SN ISSN
   DM MODE
3. $Cast no. and file no.
   CN ISTA
   FI IFIL
4. $Starting times, positions, and depths
   ST YR,DAY,RHR,RMIN,DLAT,MLAT,DLON, MLON,ZBOT
5. $Ending times, positions, and depths
   ET YR,DAY,RHR,RMIN,DLAT,MLAT,DLON, MLON,Z,ZBOT
6. $Minimum and maximum pressures expected
   MM PMIN,PMAX
7. $Sampling interval time tolerance
   DT DELTAT
8. $Pressure tolerances
   PT PL,PH,DP
9. $Temperature tolerances
   TT TL,TH,DT
10. $Conductivity tolerances
    CT CL,CH,DC
11. $Diagnostic control
    DI ISPKS
12. $Process cast
    GO
13. $Stop processing command
    QU

The dollar-sign symbol, $, indicates a comment and has no effect on the processing. A description of the input follows.

IU: Input tape unit number (default is 3).
IO: Output FEB file unit number (default is 4).
MSGW: FEB file write message level (default is 2).

ICRUIS: Cruise number (default to cruise no. on internal tape label).
SHIP: Name of ship (default KANE).
ISSN: Sensor serial number (default 0).
MODE: Deployment mode—"VPSD" for single downcast, "VPSU" for single upcast, "VPSB" for single down and upcast, "VPMD" for multiple downcasts, "VPMU" for multiple upcasts, "VPMB" for multiple down and upcasts.
ISTA: Six-digit station number (default to cast number on internal tape label).
IFIL: Input tape file number (default to next file on tape).
YR: Year.
DAY: Julian day.
TIME: Time (zulu).
DLAT: Degrees latitude.
MLAT: Minutes latitude.
DLON: Degrees longitude.
MLON: Minutes longitude.
ZBOT: Bottom depth.
PMIN: Minimum pressure expected.
PMAX: Maximum pressure expected.
DELTAT: Maximum time interval between samples (default is 0.0341 sec).
PL: Pressure tolerance lower bound (default is 0).
PH: Pressure tolerance upper bound (default is 6500).
DP: Maximum pressure difference between samples (default is 0.5).
TL: Temperature tolerance lower bound (default is −3.0).
TH: Temperature tolerance upper bound (default is 32.0).
DT: Maximum temperature difference between samples (default is 0.2).
CL: Conductivity tolerance lower bound (default is 10.0).
CH: Conductivity tolerance upper bound (default is 80.0).
DC: Maximum conductivity difference tolerance between samples (default is 0.2).
ISPKS: Diagnostics printed if ISPKS = 1, no diagnostics printed if ISPKS = 0 (default is 1).
GO: Header information has been completed, process cast.
QU: Stop processing (quit) after this instruction.

When running the program, the user will normally default many of the inputs such as input/output controls, deployment mode, diagnostic control, time tolerance, and pressure, temperature, and conductivity tolerances. Cruise and station numbers can be defaulted if the internal tape header records are correct (determined through a tape scan utility). The tape file number can be defaulted if the processing starts at file one and no files need to be skipped. Cruise number, ship, sensor serial number, and deployment mode usually are entered for only the first file to process, since these values are then maintained for remaining files until changed by another instruction. Starting/ending times, positions, and depths are necessary for each cast if the header information is to be complete. A "GO" instruction is required to begin processing on each tape file and a "QU" instruction is needed if processing is to stop before a tape double-end-of-file mark is reached. If the instruction file is empty, all casts will be processed until the tape double-end-of-file mark using defaults and internal tape label information.

Sample UNIVAC Run Stream:

```
@ASG,TJ 3.,U9S//////Q,TEM08. assign data logger tape
@MAP,I .ACTDRAW      . create executable element
IN .CTDRAW
IN .READIN
IN .PRCDAT
IN .SCREEN
IN .HEADLD
IN .BYTE2
IN .UREAD
IN .UNPACK
IN .REVBIT
IN .ZWRIT
@END
@XQT .ACTDRAW                . run the program
$ Process first three casts from data logger tape.
$ cruise no.
CR 270484
$ ship name.
SH BENT
$ station no.
CN 261005
$ starting times, positions, and depth for first station.
ST 84.,204.,22.,12.,32.,12.1,71.,18.3,3682.
$ process first station.
GO
$ station no.
CN 262006
$ starting times, positions, and depth for second station.
ST 84.,206.,01.,22.,33.,11.4,72.,44.3,3455.
$ process second station.
GO
```

```
$ station no.
CN 263007
$ starting times, positions, and depth for third station.
ST 84.,207.,11.,34.,33.,18.0,72.,55.9,3412.
$ process third cast.
GO
$ stop processing
QU
```

References

Hallock, Z. R. (1980). *The Fast and Easy Binary (FEB) Data File*. U. S. Naval Oceanographic Office, Bay St. Louis, Mississippi, Technical Note 7210-12-80.

Teague, W. J. (1984). *Conversion of FEB Utilities to ASCII FORTRAN*. Naval Ocean Research and Devlopment Activity, NSTL, Mississippi, NORDA Technical Note 275.

Appendix A: Program listings

PROGRAM LISTINGS

```
C*************************************************************************
C PROGRAM: CTDRAW
C MAIN PROGRAM FOR CONVERSION OF 9-TRACK DATALOGGER TAPES
C WRITTEN BY THE NEIL-BROWN 1150/DIGIDATA CTD SYSTEM.
C
C*************************************************************************
C
      WRITE(6,*)' RAW CTD TO FEB FORMAT PROGRAM '
C
C   READ INPUT INSTRUCTIONS
 101  CALL READIN(&99)
C
C   PROCESS THE CAST
      CALL PRCDAT(&92,&95)
C
C   READ INSTRUCTIONS FOR NEXT CAST
      GO TO 101
C
 92   WRITE(6,*)' ERROR IN PROCESSING DATA IN SUBROUTINE PRCDAT'
      STOP
C
 95   WRITE(6,*)' DOUBLE EOF REACHED'
C
 99   WRITE(6,*)' END OF JOB'
      STOP
      END
```

```
C*********************************************************************
C
C    SUBROUTINE READIN
C    THIS SUBROUTINE READS INPUT INSTRUCTIONS AND SETS DEFAULTS FOR
C    MAIN PROGRAM, CTDRAW.
C
C*********************************************************************
C
       SUBROUTINE READIN(*)
C
        IMPLICIT REAL*4 (A-H, O-Z)
        IMPLICIT INTEGER*4 (I,J,K,L,M,N)
C
        CHARACTER*4 A
        CHARACTER*6 DM
        CHARACTER*24 SHIP
        CHARACTER*12 ISSN
        CHARACTER*72 IN
C
        COMMON /COMTOL/ DELTAT,PL,PH,DP,TL,TH,DT,CL,CH,DC,ISPKS
        COMMON /COMIO/ IU,IOU,MSGW1,ICRUIS,ISTA,IFIL
        COMMON /COMTIM/ YR,DAY,RHR,RMIN,DLAT,RMLAT,DLON,
     &  RMLON,ZBOT,YR1,DAY1,RHR1,RMIN1,DLAT1,RMLAT1,DLON1,
     &  RMLON1,ZBOT1,PMIN,PMAX
        COMMON /COMALP/ SHIP,DM,ISSN
C
C    SET DEFAULTS
        DATA IU,IOU,IFIL,MSGW1 /3,4,1,2/
        DATA IPRNGE,DELTAT,ISPKS /6500,.0341,1/
        DATA PL,PH,DP /0.,6500.,0.5/
        DATA TL,TH,DT /-3.0,32.0,0.2/
        DATA CL,CH,DC /10.,80.,0.2/
        DATA SHIP,DM /'KANE','VPSD'/
C
C*********************************************************************
C
C    OPEN SCRATCH FILE TO ALLOW LIST DIRECTED READ
       CLOSE(UNIT=29)
       OPEN(UNIT=29,STATUS='SCRATCH',ERR=9096)
C
C    READ INPUT INTO INTERNAL FILE
 10    READ(5,5000,ERR=9000,END=999)IN
 5000  FORMAT(A)
C    WRITE INSTRUCTION TO UNIT 29
       WRITE(29,5000)IN(4:72)
       BACKSPACE 29
C
C*********************************************************************
C
C    TRANSLATE INSTRUCTION
       A=IN(1:2)
C
C    READ COMMENT
       IF(A(1:1).EQ.'$'.OR.A(1:1).EQ.'$')GO TO 10
```

```
C
C   READ INPUT TAPE UNIT NO.
       IF(A.EQ.'IU'.OR.A.EQ.'iu')THEN
        READ(29,*,ERR=9097)IU
        GO TO 10
       END IF
C
C   READ OUTPUT FEB FILE UNIT NO.
       IF(A.EQ.'OU'.OR.A.EQ.'ou')THEN
        READ(29,*,ERR=9097)IOU
        GO TO 10
       END IF
C
C   READ FEB FILE WRITE MESSAGE LEVEL (MSGW)
       IF(A.EQ.'MS'.OR.A.EQ.'ms')THEN
        READ(29,*,ERR=9097)MSGW1
        GO TO 10
       END IF
C
C   READ CRUISE NO.
       IF(A.EQ.'CR'.OR.A.EQ.'cr')THEN
        READ(29,*,ERR=9097)ICRUIS
        GO TO 10
       END IF
C
C   READ SHIP NAME
       IF(A.EQ.'SH'.OR.A.EQ.'sh')THEN
        SHIP=IN(4:27)
        GO TO 10
       END IF
C
C   READ SENSOR SERIAL NO.
       IF(A.EQ.'SN'.OR.A.EQ.'sn')THEN
        ISSN=IN(4:15)
        GO TO 10
       END IF
C
C   READ DEPLOYMENT MODE
       IF(A.EQ.'DM'.OR.A.EQ.'dm')THEN
        DM=IN(4:9)
        GO TO 10
       END IF
C
C   READ CAST NO.
       IF(A.EQ.'CN'.OR.A.EQ.'cn')THEN
        READ(29,*,ERR=9097)ISTA
        GO TO 10
       END IF
C
C   READ INPUT TAPE FILE NO.
       IF(A.EQ.'FI'.OR.A.EQ.'fi')THEN
        READ(29,*,ERR=9097)IFIL
        GO TO 10
       END IF
```

```
C
C   READ STARTING TIMES, POSITIONS, AND DEPTHS
      IF(A.EQ.'ST'.OR.A.EQ.'st')THEN
       READ(29,*,ERR=9097)YR,DAY,RHR,RMIN,DLAT,RMLAT,DLON,RMLON,ZBOT
       GO TO 10
      END IF
C
C   READ ENDING TIMES, POSTIONS, AND DEPTHS
      IF(A.EQ.'ET'.OR.A.EQ.'et')THEN
       READ(29,*,ERR=9097)YR1,DAY1,RHR1,RMIN1,DLAT1,RMLAT1,DLON1,RMLON1
     &  ,ZBOT1
       GO TO 10
      END IF
C
C   READ MAX TIME INTERVAL BETWEEN SAMPLES
      IF(A.EQ.'DT'.OR.A.EQ.'dt')THEN
       READ(29,*,ERR=9097)DELTAT
       GO TO 10
      END IF
C
C   READ PRESSURE TOLERANCES
      IF(A.EQ.'PT'.OR.A.EQ.'pt')THEN
       READ(29,*,ERR=9097)PL,PH,DP
       GO TO 10
      END IF
C
C   READ TEMPERATURE TOLERANCES
      IF(A.EQ.'TT'.OR.A.EQ.'tt')THEN
       READ(29,*,ERR=9097)TL,TH,DT
       GO TO 10
      END IF
C
C   READ CONDUCTIVITY TOLERANCES
      IF(A.EQ.'CT'.OR.A.EQ.'ct')THEN
       READ(29,*,ERR=9097)CL,CH,DC
       GO TO 10
      END IF
C
C   READ DIAGNOSTICS PRINTOUT CONTROL (1 FOR YES, 0 FOR NO)
      IF(A.EQ.'DI'.OR.A.EQ.'di')THEN
       READ(29,*,ERR=9097)ISPKS
       GO TO 10
      END IF
C   READ MIN AND MAX PRESSURES EXPECTED
      IF(A.EQ.'MM'.OR.A.EQ.'mm')THEN
       READ(29,*,ERR=9097)PMIN,PMAX
       GO TO 10
      END IF
C
C   PROCESS CAST
      IF(A.EQ.'GO'.OR.A.EQ.'go')RETURN
C
C   STOP PROCESSING
      IF(A.EQ.'QU'.OR.A.EQ.'qu')THEN
```

```
        WRITE(6,*)'PROCESSING TERMINATED ON "QU" COMMAND'
        RETURN1
       END IF
C
C
C*****************************************************************************
C
 9000  WRITE(6,*)'ERROR IN READING INSTRUCTION, TRY AGAIN'
       GO TO 10
 9096  WRITE(6,*)'ERROR IN OPENING UNIT 29, DO NOT USE UNIT 29!'
       STOP
 9097  WRITE(6,*)'ERROR WHILE READING INSTRUCTION ',A,' TRY AGAIN'
       GO TO 10
  999  RETURN
       END
```

```
C*************************************************************************
C
C   SUBROUTINE:  PRCDAT
C   THIS PROGRAM PROCESSES THE RAW CTD DATA AND WRITES DATA
C   TO FEB FORMAT
C
C*************************************************************************
C
      SUBROUTINE PRCDAT(*,*)
C
        IMPLICIT REAL*4 (A-H,O-Z)
        IMPLICIT INTEGER*4 (I,J,K,L,M,N)
C
        CHARACTER*6 DM
        CHARACTER*12 ISSN
        CHARACTER*24 SHIP
        CHARACTER*6 ADOCW,IPW,NMFW,NMBW,PNW,RNW
C
        COMMON /COMIO/ IU,IOU,MSGW1,ICRUIS,ISTA,IFIL
        COMMON /COMTOL/ DELTAT,PL,PH,DP,TL,TH,DT,CL,CH,DC,ISPKS
        COMMON /COMALP/ SHIP,DM,ISSN
C
        COMMON /HEDDAT/ KPCODE,KTCAST,KDAY,KSTA,KCRUIS
C
        COMMON /ENGDAT/ PP(170),TT(170),CC(170),HH(170)
C
C   ZWRIT COMMON AREAS
      COMMON / WHDR /LW,NW,NBW,NFW,NIW,NAW
      COMMON / WHDR1 /NMBW,NMFW,PNW,RNW,IPW(4)
      COMMON / WDOCF /FDOCW(25) /WDOCI/IDOCW(15) /WDOCA/ADOCW(72)
      COMMON / WDATA / VW(4,1000)
      COMMON / DIAGS / MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
C
      REAL*4 MINMAX(6)
C
      DATA NNNW,NNIP,NNF,NNI,NNA/1000,4,25,15,72/
      DATA IPW/'RELSEC','PRESS ','TEMP  ','COND  '/
      DATA IOPEN /0/
      DATA ISCAN /0/
C
C*************************************************************************
      MSGW=MSGW1
      JFIL=IFIL
      IUNIT=IU
      ISW=1
      ISEG=0
      IDOCW1=1
      LW=4
C
 101  NBWP1=NBW+1
C
      DO 200 I=1,1000
      NW=I
C
```

```
C   GET A DATA RECORD
C   IF IEOF=0 THEN DATA RECORD IS RETURNED
C   IF IEOF=1 THEN END OF FILE REACHED
C   IF IEOF=2 THEN DOUBLE END OF FILE REACHED
C
      IF(ISCAN.EQ.0)CALL UNPACK(IUNIT,JFIL,IEOF)
      IF(IEOF.GE.1)ISCAN=0
C
      IF(IEOF.EQ.1)THEN
       IOPEN=0
C   CHECK TO SEE IF ANY DATA HAS BEEN LOADED TO VW
       IF(NW.EQ.1)GO TO 211
       GO TO 210
      END IF
C
      IF(IEOF.EQ.2)RETURN2
C
C
      ISCAN=ISCAN+1
      P=PP(ISCAN)
      T=TT(ISCAN)
      C=CC(ISCAN)
      H=HH(ISCAN)
C
      IF(ISCAN.EQ.170)ISCAN=0
C
CC
C   CHECK HEADER RECORD
      IF(ISW.EQ.1)THEN
       ISW=0
C
        IF(ISTA.NE.0)THEN
         WRITE(NMBW,6010)ISTA
        ELSE
         WRITE(NMBW,6010)KSTA
        END IF
        IF(ICRUIS.NE.0)THEN
         WRITE(NMFW,6010)ICRUIS
        ELSE
         WRITE(NMFW,6010)KCRUIS
 6010    FORMAT(I6)
        END IF
C
       REFTIM=KDAY
C
C   OPEN FEB    E AND LOG FILE
       IF(IOPEN.EQ.0)THEN
         CLOSE(UNIT=28)
         CLOSE(UNIT=IOU)
         IOPEN=1
         JSEG=1
         IF(ISPKS.EQ.1)OPEN(UNIT=28,FILE=NMBW//'LOG',STATUS='NEW',
     *   ERR=9094)
C   NOTE: IF RECL=IBUF IN ZWRIT CHANGES, RECL HERE MUST BE CHANGED.
```

```
C   RCDS IS SPECIFIED SO THAT THE UNIVAC FILE CAN BE LARGER THAN THE
C   DEFAULT TRACK LIMIT OF 128; 15000 ALLOWS FOR ABOUT 5000 TRACKS.
C   RCDS IS SPECIFIC TO THE UNIVAC AND IS NOT USED ON THE VAX.
C   THE FOLLOWING OPEN STATEMENT IS FOR THE VAX
        OPEN(UNIT=IOU,FILE='RCT'//NMBW,FORM='UNFORMATTED',
     *       ACCESS='DIRECT',RECL=600,STATUS='NEW',ERR=9095)
       WRITE(6,*)
       WRITE(6,*)'WRITE FEB FILE: ','RCT'//NMBW
       WRITE(28,*)'FEB FILE: ','RCT'//NMBW
       WRITE(6,*)
C THE FOLLOWING OPEN STATEMENT IS FOR THE UNIVAC
C       OPEN(UNIT=IOU,FILE='RCTD'//NMFW//'*'//NMBW,FORM='UNFORMATTED',
C    *       ACCESS='DIRECT',RECL=600,STATUS='NEW',RCDS=15000,
C    *       ERR=9095)
C
C       WRITE(6,*)
C       WRITE(6,*)'WRITE FEB FILE: ','RCTD'//NMFW//'*'//NMBW
C       WRITE(28,*)'FEB FILE: ','RCTD'//NMFW//'*'//NMBW
C       WRITE(6,*)
      END IF
C
      WRITE(6,2000)REFTIM
 2000 FORMAT(//' REFERENCE TIME FROM HEADER =',G11.5,' DAYS'//)
C
      IF(KCRUIS.NE.ICRUIS)WRITE(6,2100)ICRUIS,KCRUIS
 2100 FORMAT(/' CRUISE NUMBERS DISAGREE. SUPPLIED: ',I6,
     *   '   ON LOGGER TAPE: ',I6//)
C
C   CHECK CAST NO. (I.E. LAST 3 DIGITS)
      ISTA3=MOD(ISTA,1000)
      IF(KSTA.NE.ISTA3)WRITE(6,2101)ISTA3,KSTA
 2101 FORMAT(/' CAST NUMBERS DISAGREE. SUPPLIED: ',I6,
     *   '   ON LOGGER TAPE: ',I6//)
C
     END IF
CC
C
C   LOAD DATA INTO FEB COMMON
      VW(1,I)=H
      VW(2,I)=P
      VW(3,I)=T
      VW(4,I)=C
C
C   SCREEN THE DATA
      CALL SCREEN(NBWP1,I,H,P,T,C,MINMAX,NTDIAG,NDDIAG)
      PMAX=MINMAX(2)
C
 200  CONTINUE
C
 210  ISEG=ISEG+1
C
C   UPDATE END OF SERIES FLAG AND REL. PROF. NO.
      IDOCW(1)=IEOF
      NBW1=NBW+1
```

```
      WRITE(RNW,2110)NBW1
 2110 FORMAT(I6)
C   WRITE FULL HEADER RECORD ONLY FOR FIRST SEGMENT
      IF(IDOCW1.EQ.1)THEN
       NFW=25
       NIW=15
       NAW=72
C   LOAD THE HEADERS INTO FEB COMMON
       CALL HEADLD
      ELSE
       NFW=0
       NIW=1
       NAW=0
      END IF
C
C   WRITE THE DATA TO A FEB FILE
      CALL ZWRIT(IOU,IF,JSEG)
C   ZWRIT ERROR RETURN
      IF(IF.NE.0) RETURN1
      IDOCW1=IDOCW(1)
      JSEG=JSEG+1
C
C   GET MORE DATA
      IF(IEOF.EQ.0)GO TO 101
C
C   ON END OF FILE WRITE CAST SUMMARY INFORMATION
  211 WRITE(6,6100)MINMAX
 6100 FORMAT(//' CAST COMPLETE.'//
     *   '   PMIN=',G12.6,';   PMAX=',G12.6//
     *   '   TMIN=',G12.6,';   TMAX=',G12.6//
     *   '   CMIN=',G12.6,';   CMAX=',G12.6//)
      WRITE(6,*)
      WRITE(6,*)' NO. OF DIAGNOSTICS'
      WRITE(6,*)' TIME: ',NTDIAG
      WRITE(6,*)' COND., TEMP., AND DEPTH: ',NDDIAG
C
      RETURN
 9094 WRITE(6,*)'ERROR IN OPENING LOG FILE, UNIT 28'
      STOP
 9095 WRITE(6,*)'ERROR IN OPENING FEB FILE, UNIT ',IOU
      STOP
      END
```

```
C*****************************************************************************
C
C   SUBROUTINE SCREEN
C   THIS ROUTINE CHECKS FOR WILD POINTS IN P, T, C, AND TIME, AND COUNTS
C   THE NO. OF WILD POINTS (ITDIAG FOR TIME, AND IDDIAG FOR P, T, AND C).
C
C*****************************************************************************
C
      SUBROUTINE SCREEN(NBWZ,KCYC,TYME,PRES,TEMP,COND,
     *   MINMAX,ITDIAG,IDDIAG)
C
      IMPLICIT REAL*4 (A-H,O-Z)
      IMPLICIT INTEGER*4 (I,J,K,L,M,N)
C
      CHARACTER*2 JFLG(6)
C
      COMMON /COMTOL/ DELTAT,PL,PH,DP,TL,TH,DT,CL,CH,DC,ISPKS
C
      REAL*4 MINMAX(6)
C
        ISSW=1
        TIMTOL=DELTAT
        DO 1 I=1,6
    1  JFLG(I)='  '
C
        IF((KCYC.LT.3).AND.(NBWZ.LT.2)) THEN
C
        PPRES=PRES
        PTEMP=TEMP
        PCOND=COND
        PTYME=TYME
        PMIN=10000.
        PMAX=-10.
        TMIN=100.
        TMAX=-10.
        CMIN=100.
        CMAX=-10.
C
        END IF
C
        IF(PRES.LT.PL) JFLG(1)='LO'
        IF(PRES.GT.PH) JFLG(1)='HI'
        IF(ABS(PRES-PPRES).GT.DP) JFLG(2)='CH'
C
        IF(TEMP.LT.TL) JFLG(3)='LO'
        IF(TEMP.GT.TH) JFLG(3)='HI'
        IF(ABS(TEMP-PTEMP).GT.DT) JFLG(4)='CH'
C
        IF(COND.LT.CL) JFLG(5)='LO'
        IF(COND.GT.CH) JFLG(5)='HI'
        IF(ABS(COND-PCOND).GT.DC) JFLG(6)='CH'
C
        DO 2 J=1,6
    2 IF(JFLG(J).NE.'  ') ISSW=2
```

```
C
        TMDIFF=TYME-PTYME
        KCYCP=KCYC-1
C
       IF(TMDIFF.GT.TIMTOL)THEN
       IF(ISPKS.EQ.1)WRITE(28,6011) NBWZ,KCYCP,
      *PTYME,KCYC,TYME,TMDIFF,TIMTOL
 6011    FORMAT(' TIME DIFF. IN SEG. ',
     *I4,' BETWEEN CY.',I4,' TIME=',F12.4,' AND CY.',
     *I4,' TIME=',F12.4,' DIFF.=',F8.4,' MAX=',F6.4,T132,'$')
C
       ITDIAG=ITDIAG+1
C
       END IF
C
         IF(ISSW.EQ.1) THEN
         PPRES=PRES
         PTEMP= TEMP
         PCOND= COND
         PTYME=TYME
C
         IF(PRES.GT.PMAX) PMAX=PRES
         IF(PRES.LT.PMIN) PMIN=PRES
         IF(TEMP.GT.TMAX) TMAX=TEMP
         IF(TEMP.LT.TMIN) TMIN=TEMP
         IF(COND.GT.CMAX) CMAX=COND
         IF(COND.LT.CMIN) CMIN=COND
C
         MINMAX(1)=PMIN
         MINMAX(2)=PMAX
         MINMAX(3)=TMIN
         MINMAX(4)=TMAX
         MINMAX(5)=CMIN
         MINMAX(6)=CMAX
C
         RETURN
C
         END IF
C
         PDIFF=PRES-PPRES
         TDIFF=TEMP-PTEMP
         CDIFF=COND-PCOND
C
         IF(ISPKS.EQ.1)WRITE(28,6) NBWZ,KCYC,JFLG(1),JFLG(2),PRES,PDIFF,
      *JFLG(3),JFLG(4),TEMP,TDIFF,JFLG(5),JFLG(6),COND,CDIFF
C
     6 FORMAT(' SEG ',I4,' CYCLE ',I4,3X,A2,'/',A2,'.PRES=',
      *G12.6,'/',G10.4,3X,A2,'/',A2,'.TEMP=',G12.6,'/',
      *G10.4,3X,A2,'/',A2,'.COND=',G12.6,'/',G10.4,T132,'$')
C
         IDDIAG=IDDIAG+1
C
C
         PPRES=PRES
```

```
      PTEMP=TEMP
      PCOND=COND
      PTYME=TYME
C
      RETURN
      END
```

```
C*******************************************************************************
C
C   SUBROUTINE HEADLD
C   THIS ROUTINE LOADS FEB HEADER COMMON AREAS.
C
C*******************************************************************************
C
      SUBROUTINE HEADLD
C
C
        IMPLICIT REAL*4 (A-H,O-Z)
        IMPLICIT INTEGER*4 (I,J,K,L,M,N)
C
        CHARACTER*6 ADOCW,IPW,NMFW,NMBW,PNW,RNW
        CHARACTER*6 DM
        CHARACTER*24 SHIP
        CHARACTER*12 ISSN
        CHARACTER*8 MDY,ITIM
C
        COMMON /COMIO/ IU,IOU,MSGW1,ICRUIS,ISTA,IFIL
        COMMON /COMTOL/ DELTAT,PL,PH,DP,TL,TH,DT,CL,CH,DC,ISPKS
        COMMON /COMTIM/ YR1,DAY1,RHR1,RMIN1,DLAT1,RMLAT1,DLON1,
     &  RMLON1,ZBOT1,YR2,DAY2,RHR2,RMIN2,DLAT2,RMLAT2,DLON2,
     &  RMLON2,ZBOT2,PMIN,PMAX
        COMMON /COMALP/ SHIP,DM,ISSN
C
C   ZWRIT COMMON AREAS
C
      COMMON / WHDR /LW,NW,NBW,NFW,NIW,NAW
      COMMON / WHDR1 /NMBW,NMFW,PNW,RNW,IPW(4)
      COMMON /WDOCF/FDOCW(25) /WDOCI/IDOCW(15) /WDOCA/ADOCW(72)
C
      DATA ADOCW /72*'      '/
C
C*******************************************************************************
C
      PNW='      '
C
      FDOCW(1)=YR1
      FDOCW(2)=DAY1
      FDOCW(3)=RHR1
      FDOCW(4)=RMIN1
      FDOCW(5)=DLAT1
      FDOCW(6)=RMLAT1
      FDOCW(7)=DLON1
      FDOCW(8)=RMLON1
      FDOCW(9)=ZBOT1
      FDOCW(10)=YR2
      FDOCW(11)=DAY2
      FDOCW(12)=RHR2
      FDOCW(13)=RMIN2
      FDOCW(14)=DLAT2
      FDOCW(15)=RMLAT2
      FDOCW(16)=DLON2
```

```
      FDOCW(17)=RMLON2
      FDOCW(18)=ZBOT2
      FDOCW(19)=PMIN
      FDOCW(20)=PMAX
      FDOCW(21)=.032
      FDOCW(22)=.032
      FDOCW(23)=DELTAT
C
      IDOCW(2)=ICRUIS
      IDOCW(3)=ISTA
      IDOCW(11)=6500
C
      ADOCW(1)=SHIP(1:6)
      ADOCW(2)=SHIP(7:12)
      ADOCW(3)=SHIP(13:18)
      ADOCW(4)=SHIP(19:24)
      ADOCW(5)='DB'
      ADOCW(6)='SEC'
      ADOCW(7)='SEC'
      ADOCW(58)=DM
      ADOCW(59)='CTD'
      ADOCW(60)='NB3'
      ADOCW(61)=ISSN(1:6)
      ADOCW(62)=ISSN(7:12)
C   UNIVAC DATE ROUTINE
C     CALL ADATE(MDY,ITIM)
C     ADOCW(66)=MDY(1:6)
C   END UNIVAC DATE SETUP
C   HODAS DATE ROUTINE
C     CALL ?
C   VAX DATE ROUTINE
      CALL IDATE(JMO,JDA,JYE)
      WRITE(ADOCW(66),600)JMO,JDA,JYE
  600 FORMAT(3I2)
C   END VAX DATE SETUP
C
      RETURN
      END
```

```
C*************************************************************************
C
C       SUBROUTINE BYTE2
C       THIS SUBROUTINE SPLITS AN 8 BIT BYTE INTO TWO 4 BIT BYTES.
C       IT RETURNS THE LEFT 4 BITS (IL) AND THE RIGHT 4 BITS (IR).
C
C*************************************************************************
        SUBROUTINE BYTE2(IBYTE,IL,IR)
C
        IMPLICIT REAL*4 (A-H,O-Z)
        IMPLICIT INTEGER*4 (I,J,K,L,M,N)
C
        IN=16
        IL=IBYTE/IN
        IR=IBYTE-IL*IN
        RETURN
        END
```

```
C*****************************************************************************
C
C    SUBROUTINE RBYTE
C    THIS SUBROUTINE READS THE CTD DATA LOGGER TAPE (NAVOCEANO FORMAT) AND
C    RETURNS 2040 DATA BYTES (STORED 1 BYTE PER WORD).
C    THIS SUBROUTINE IS DESIGNED FOR USEAGE ON A VAX 11/750.
C
C*****************************************************************************
C
      SUBROUTINE RBYTE(IUNIT,IFIL,JREAD,IFLG)
C
      IMPLICIT REAL*4 (A-H,O-Z)
      IMPLICIT INTEGER*4 (I,J,K,L,M,N)
C
      BYTE IREAD(2044)
      DIMENSION JREAD(1)
      INTEGER*2 IUNIT,NIREAD
      DATA NIREAD,ICK /2043,0/
C    NIREAD IS SET TO 2043 SINCE THE TAPE IS COPIED VIA THE UNIVAC, A 36-BIT
C    WORD MACHINE (2043 X 8 IS DIVISIBLE BY 36).
C
C    SET TAPE CHARACTERISTICS FOR INITIAL READ
      IF(ICK.EQ.0)THEN
       CALL TCHAR(IUNIT,'MSA0:','BINARY','1600','ODD')
       ICK=1
C
C    POSITION TAPE TO STARTING FILE
       ISKIP=IFIL-1
       IF(ISKIP.GT.0)THEN
        CALL TFILESKIP(IUNIT,ISKIP,ISTAT)
          IF(ISTAT.NE.1)THEN
           WRITE(6,*)' PROBLEM IN POSITIONING TAPE UNIT',IUNIT
           STOP
          END IF
       END IF
C
      END IF
C
C    READ TAPE RECORD
      CALL TREAD(IUNIT,IREAD,NIREAD,ISTAT)
C
C    CHECK STATUS OF READ
C    SET STATUS FLAG: IFLG=0 FOR DATA READ, =1 FOR EOF, =2 FOR ERROR.
      IFLG=0
C    CHECK FOR END OF FILE
      IF(ISTAT.EQ.2160)THEN
       IFLG=1
       RETURN
      END IF
C    CHECK FOR PROBLEM DURING READ
      IF(NIREAD.LE.0)IFLG=2
      IF(ISTAT.NE.1)THEN
       IFLG=2
       RETURN
```

```
      END IF
C
C   BYTE RANGE ON VAX IS -127 TO +128
C   WANT RANGE OF 0 TO +255, SO ADD 256 TO NEGATIVE DATA BYTES
      DO IJ=1,NIREAD
      JREAD(IJ)=IREAD(IJ)
      IF(JREAD(IJ).LT.0)JREAD(IJ)=JREAD(IJ)+256
      END DO
C
      RETURN
      END
```

```
C*********************************************************************
C
C    SUBROUTINE UREAD
C    THIS SUBROUTINE READS THE CTD DATA LOGGER TAPE (NAVOCEANO FORMAT)
C    AND RETURNS 2040 DATA BYTES (STORED 1 BYTE PER WORD).
C    THIS SUBROUTINE IS DESIGNED FOR USEAGE ON A UNIVAC 1180.
C
C*********************************************************************
C
      SUBROUTINE UREAD(IUNIT,IFIL,JREAD,IFLG)
C
       IMPLICIT REAL*4 (A-H,O-Z)
       IMPLICIT INTEGER*4 (I,J,K,L,M,N)
C
       DIMENSION IREAD(510),IBYTE(4),JREAD(2044)
C
      DATA NIREAD,IEOF/510,1/
C
C
C    POSITION TAPE BY FILE
      IF(IEOF.EQ.1)THEN
       ISKIP=IFIL-1
      IEOF=0
       IF(ISKIP.GT.0)CALL NTRAN$(IUNIT,8,ISKIP)
      END IF
C
C    READ TAPE RECORD
         CALL NTRAN$(IUNIT,3,NIREAD,IREAD,IST,22)
C     IGNORE UNIVAC WARNING 1806, NO. OF ARGUMENTS IS CORRECT
C
C    CHECK STATUS OF READ
C    SET STATUS FLAG:  IFLG=0 FOR DATA READ, =1 FOR EOF, =2 FOR ERROR
      IFLG=0
C    CHECK FOR END OF FILE
      IF(IST.EQ.-2)THEN
       IFLG=1
      IEOF=1
       RETURN
      END IF
C    CHECK FOR PROBLEM DURING READ
      IF(NIREAD.LE.0)THEN
       IFLG=2
       RETURN
      END IF
      IF(IST.LT.-2)THEN
       IFLG=2
       RETURN
      END IF
C
C    SEPARATE BYTES INTO 1 PER WORD
C
       LCT=1
       IN=512
       N=1
```

```
 3      ICT=ICT+1
C    RESET COUNTER AT END OF TAPE BLOCK
C
       IF(ICT.EQ.511)THEN
       ICT=0
       RETURN
C
       END IF
       IVAR=IREAD(ICT)
 1     CONTINUE
       I11=IVAR/IN
       I1=I11*IN
       IBYTE(N)=IVAR-I1
       IVAR=I11
       N=N+1
C    LOAD EACH BYTE INTO JREAD
       IF(N.EQ.5)THEN
      JREAD(LCT)=IBYTE(4)
      JREAD(LCT+1)=IBYTE(3)
      JREAD(LCT+2)=IBYTE(2)
      JREAD(LCT+3)=IBYTE(1)
       N=1
       LCT=LCT+4
       GO TO 3
       END IF
C
       GO TO 1
C
       END
```

```
C*******************************************************************************
C
C   SUBROUTINE REVBIT
C   THIS SUBROUTINE REVERSES THE BITS IN AN 8-BIT BYTE.
C
C*******************************************************************************
C
      SUBROUTINE REVBIT(INUM,MUNI)
C
      IMPLICIT REAL*4 (A-H,O-Z)
      IMPLICIT INTEGER*4 (I,J,K,L,M,N)
C
      DIMENSION KB(16),JREV(256)
C
C   KB CONTAINS THE VALUE OF A 4 BIT BYTE FOR I=1-15,
C   AFTER THE BITS OF I HAVE BEEN REVERSED.
C   EX. I=1 (0001) THEN KB(1)=8 (1000).
      DATA KB /0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15/
C
      DATA IPASS /0/
C
C   CALCULATE THE VALUE OF AN 8 BIT BYTE UPON REVERSING
C   THE BITS.  THIS NEEDS TO BE DONE ONLY ONCE, SAVE IN JREV.
C   TECHNIQUE:  SPLIT 8 BIT BYTE INTO TWO 4 BIT BYTES AND
C   LOOK UP IN KB THE REVERSE OF THE 4 BIT BYTES.  THEN
C   RECOMBINE INTO AN 8 BIT BYTE (REVERSED).
      IF(IPASS.EQ.0)THEN
      IPASS=1
      JREV(1)=0
      DO 10 I=1,255
      CALL BYTE2(I,IL,IR)
 10   JREV(I+1)=KB(IR+1)*16 + KB(IL+1)
      END IF
C
      IF(INUM.LT.0.OR.INUM.GT.255)GO TO 900
      MUNI=JREV(INUM+1)
C
      RETURN
 900  WRITE(6,*)' PROBLEM IN REVBIT SUBROUTINE'
      END
```

```
C*******************************************************************************
C
C   SUBROUTINE UNPACK
C   THIS SUBROUTINE UNPACKS THE HEADER AND DATA BLOCKS, RETURNING THE HEADER
C   INFORMATION AND DATA RECORDS IN ENGINEERING UNITS.
C
C*******************************************************************************
C
      SUBROUTINE UNPACK(IUNIT,IFIL,IEOF)
C
      IMPLICIT REAL*4 (A-H,O-Z)
      IMPLICIT INTEGER*4 (I,J,K,L,M,N)
C
      COMMON /ENGDAT/ PP(170),TT(170),CC(170),HH(170)
      COMMON /HEDDAT/ IPCODE,ITCAST,JDAY,ISTA,ICRUIS
      DIMENSION IBUF(2044)
      DATA IREC,IM /0,256/
C
C*******************************************************************************
C
C   READ A TAPE RECORD
C
C   FOR VAX
 10   CALL RBYTE(IUNIT,IFIL,IBUF,IFLG)
C
C   FOR UNIVAC
C10   CALL UREAD(IUNIT,IFIL,IBUF,IFLG)
C
C   COUNT TAPE BLOCKS READ
      IREC=IREC+1
C
C   CHECK STATUS OF BLOCK READ
C   IFLG=0 FOR DATA BLOCK READ, IFLG=1 FOR END OF FILE READ,
C       =2 FOR PROBLEM
      IF(IFLG.EQ.1)THEN
C
C   SET COUNTER TO CHECK FOR DOUBLE EOF, IF TRUE STOP
       IEOF=IEOF+1
       IF(IEOF.EQ.2)RETURN
C
       WRITE(6,*)'END OF FILE',IFIL,' REACHED AT BLOCK',IREC
       IFIL=IFIL+1
       IREC=0
       RETURN
      END IF
C
      IEOF=0
C
      IF(IFLG.EQ.2)THEN
       WRITE(6,*)'PROBLEM ENCOUNTERED WHILE READING RECORD',IREC
       STOP
      END IF
C
C*******************************************************************************
```

```
C
C    BYTE 1: FRAME SYNC, 15 OR 240 FOR GOOD DATA; 0 OR 255 FOR HEADER.
C    IF HEADER, THE 8-BIT BYTES ARE SPLIT INTO 4-BIT BYTES AND DECODED,
C               SEE CODE FOR DECODING HEADER BELOW.
C    DATA BYTES
C    BYTE 2: LEAST SIGNIFICANT PRESSURE
C    BYTE 3: MOST SIGNIFICANT PRESSURE
C    BYTE 4: LEAST SIGNIFICANT TEMPERATURE
C    BYTE 5: MOST SIGNIFICANT TEMPERATURE
C    BYTE 6: LEAST SIGNIFICANT CONDUCTIVITY
C    BYTE 7: MOST SIGNIFICANT CONDUCTIVITY
C    BYTE 8: SIGN BYTE, BIT 1 FOR PRESSURE, BIT 2 FOR TEMPERATURE
C    BYTES 9-12: TIME BYTES
C
C*******************************************************************************
C
C    CHECK FOR HEADER RECORD
       IF(IBUF(1).EQ.0.OR.IBUF(1).EQ.255)THEN
C
C    DECODE HEADER RECORD: PRES CODE, TYPE CAST, JUL. DAY, STATION, CRUISE NO.
C
         CALL BYTE2(IBUF(2),IL,IR)
         IPCODE=IL
         ITCAST=IR
C
         CALL BYTE2(IBUF(3),IL,IR)
         CALL BYTE2(IBUF(4),IL1,IR1)
         JDAY=IR1*100 + IL*10 + IR
C
         CALL BYTE2(IBUF(5),IL,IR)
         ISTA=IL*100 + IR*10 + IL1
C
         CALL BYTE2(IBUF(6),IL,IR)
         CALL BYTE2(IBUF(7),IL1,IR1)
         CALL BYTE2(IBUF(8),IL2,IR2)
         ICRUIS=IL2*10**5+IR2*10**4+IL1*1000+IR1*100+IL*10+IR
C
         WRITE(6,*)
         WRITE(6,*)'                    TAPE HEADER CONTENTS'
         WRITE(6,*)'         (PCODE, TYPE CAST, DAY, STATION, CRUISE)'
         WRITE(6,*)IPCODE,ITCAST,JDAY,ISTA,ICRUIS
         GO TO 10
         ELSE
C
C
C        WRITE(6,*)(IBUF(I),I=1,12)
         L=0
         DO 200 J=1,2040,12
          L=L+1
C    GET PRESSURE SIGN
          IPS=1
          IF(IBUF(J+7).EQ.1.OR.IBUF(J+7).EQ.3)IPS=-1
C    GET TEMPERATURE SIGN
          ITS=1
```

```
      IF(IBUF(J+7).EQ.2.OR.IBUF(J+7).EQ.3)ITS=-1
C   CALCULATE PRES, TEMP, AND COND
      PP(L)=IPS*(IBUF(J+2)*IM + IBUF(J+1))/10.
      TT(L)=ITS*(IBUF(J+4)*IM + IBUF(J+3))/2000.
      CC(L)=(IBUF(J+6)*IM + IBUF(J+5))/1000.
C  CALCULATE TIME
C  FIRST REVERSE BITS IN EACH TIME BYTE
      CALL REVBIT(IBUF(J+8),IREV9)
      CALL REVBIT(IBUF(J+9),IREV10)
      CALL REVBIT(IBUF(J+10),IREV11)
      CALL REVBIT(IBUF(J+11),IREV12)
C   ADD BYTES TOGETHER
      LCT=LCT+1
      HH(L)=(IREV12*IM**3 + IREV11*IM**2 + IREV10*IM + IREV9)/1000.
CW      WRITE(6,*)'HH,PP,TT,CC,L',HH(L),PP(L),TT(L),CC(L),L
 200  CONTINUE
C
      END IF
C
      RETURN
      END
```

```
      SUBROUTINE ZWRIT(JU,IF,IBL)
C
C  THIS SUBROUTINE IS THE WRITE HALF OF AN INPUT-OUTPUT
C  PACKAGE FOR HANDLING NON-FORMATTED, ASCII FORTRAN
C  WRITTEN DATA FILES, COMMONLY REFERRED TO AS
C  FEB (FAST & EASY BINARY) FILES.
C
C
        IMPLICIT REAL*4 (A-H,O-Z)
        IMPLICIT INTEGER*4 (I,J,K,L,M,N)
C
      CHARACTER*6 ADOCR,IPR,NMFR,NMBR,PNR,RNR
      CHARACTER*6 ADOCW,IPW,NMFW,NMBW,PNW,RNW
      COMMON / WHDR /LW,NW,NBW,NFW,NIW,NAW
      COMMON / WHDR1 /NMBW,NMFW,PNW,RNW,IPW(1)
      COMMON /WDOCF/FDOCW(1) /WDOCI/IDOCW(1) /WDOCA/ADOCW(1)
      COMMON / WDATA / VW(1)
C
C      NOTE:  ORIGINAL ZREAD CONSISTED OF RHDR AND WHDR COMMONS ALONE,
C      ASCII FORTRAN REQUIRED PLACING CHARACTER VARIABLES IN
C      SEPARATE COMMONS - RHDR1 AND WHDR1.
C
      COMMON /RHDR /LR,NR,NBR,NFR,NIR,NAR
      COMMON /RHDR1 /NMBR,NMFR,PNR,RNR,IPR(1)
      COMMON /RDOCF/FDOCR(1) /RDOCI/IDOCR(1) /RDOCA/ADOCR(1)
      COMMON / RDATA / VR(1)
C
C
      COMMON / DIAGS / MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
      LOGICAL B1,B210,B10,B35,B45,B69,OD
      COMMON / JPOS / JUNIT(99)
      COMMON /DRDCOM/ JFLG,ISECR(99)
      DIMENSION IUNIT(99)
      DATA MSGW / 2 /
      DATA LLSW / 1 /, IRST, IWST / 1, 1 /
      DATA IBUF/600/
C
      IW=1
      IF(JU.LT.0) IW=2
      IU=ABS(JU)
C
C      OPEN DIRECT ACCESS FEB FILES AS REQUIRED.
C      RECORD SIZE IS SET TO IBUF IN DATA STATEMENT.
C      EACH HOLLERITH WORD CONSISTS OF SIX CHARACTERS,
C      AND THUS OCCUPIES ONE AND ONE-HALF WORDS.
C
      IF(IU.NE.IUSAV)THEN
       INQUIRE(UNIT=IU,OPENED=OD)
       IF(.NOT.OD) OPEN(UNIT=IU,ACCESS='DIRECT',FORM='UNFORMATTED',
     &   STATUS='UNKNOWN',ERR=9090,RECL=IBUF)
       IUSAV=IU
      END IF
C
C
```

```
      B1=MSGW.EQ.1
      B210=MSGW.GE.2.AND.MSGW.LE.10
      B10=MSGW.EQ.10
      B35=MSGW.EQ.3.OR.MSGW.EQ.5.OR.MSGW.EQ.7.OR.MSGW.EQ.9.OR.MSGW.EQ.10
      B45=MSGW.EQ.4.OR.MSGW.EQ.5.OR.MSGW.GE.8.AND.MSGW.LE.10
      B69=MSGW.GE.6.AND.MSGW.LE.9
C
      IBLK=IBL
      IPOS=JUNIT(IU)
      IREC=ISECR(IU)
      IF(IPOS.EQ.0) IPOS=1
      IF(IREC.EQ.0) IREC=1
      IF(IBL.EQ.0) IBLK=IUNIT(IU)
      IF(IBLK.LT.IPOS) GO TO 5
    4 IF(IBLK.EQ.IPOS) GO TO 3
C
C      FULL DUMMY READ IS REQUIRED IN ORDER TO VERIFY RECORD.
C      FILES ARE ZERO-FILLED IN INITIAIZATION ON THE UNIVAC,
C      BUT ARE NOT ON THE VAX - ERR=99 BRANCH IS USED.
      READ(IU'IREC,ERR=99)LQ,NQ,NFQ,NIQ,NAQ,NBQ,(NMBQ,I=1,LQ+4),
     &(FDOCQ,I=1,NFQ),(IDOCQ,I=1,NIQ),(NMBQ,I=1,NAQ),
     &(VQ,M=((IRST-1)*LQ+1),(IRST-1)*LQ+IBUF
     &-(10+LQ+NFQ+NIQ+NAQ+(5+LQ+NAQ)/2))
      IF(LQ.EQ.0.OR.NBQ.EQ.0.OR.NFQ.LT.0.OR.NIQ.LT.0.OR.NAQ.LT.0)
     &GO TO 99
C       IWORDS IS THE TOTAL WORDS CONTAINED IN THE SEGMENT
      IWORDS=(10+LQ+NFQ+NIQ+NAQ+(5+LQ+NAQ)/2)+(LQ*NQ)
C       IREC IS THE RECORD NO. FOR THE NEXT SEGMENT
      IREC=((IWORDS-1)/IBUF)+1+IREC
      IPOS=IPOS+1
      IUNIT(IU)=IPOS
      JUNIT(IU)=IPOS
      GO TO 4
C
  5     IREC=1
      IPOS=1
      ISECR(IU)=0
      IUNIT(IU)=IPOS
      JUNIT(IU)=IPOS
      IF (IBL.NE.0) GO TO 4
C
C      FIND RECORD NO. FOR WRITE AT END OF FILE, IBL=0
  2     READ(IU'IREC,ERR=6)LQ,NQ,NFQ,NIQ,NAQ,NBQ,(NMBQ,I=1,LQ+4),
     &(FDOCQ,I=1,NFQ),(IDOCQ,I=1,NIQ),(NMBQ,I=1,NAQ),
     &(VQ,M=((IRST-1)*LQ+1),(IRST-1)*LQ+IBUF
     &-(10+LQ+NFQ+NIQ+NAQ+(5+LQ+NAQ)/2))
      IF(LQ.EQ.0.OR.NBQ.EQ.0.OR.NFQ.LT.0.OR.NIQ.LT.0.OR.NAQ.LT.0)
     &GO TO 6
      IWORDS=(10+LQ+NFQ+NIQ+NAQ+(5+LQ+NAQ)/2)+(LQ*NQ)
      IREC=((IWORDS-1)/IBUF)+1+IREC
      IPOS=IPOS+1
      IUNIT(IU)=IPOS
      JUNIT(IU)=IPOS
C2
```

```
      GO TO 2
C
    6 CONTINUE
      WRITE(6,1001)IU
C
C
    3 CONTINUE
      NBW=IPOS
      GO TO (81,82),IW
   81 IF(NW.GT.NNNW.OR.LW.GT.NNIP.OR.NFW.GT.NNF.
     *   OR.NIW.GT.NNI.OR.NAW.GT.NNA) GO TO 95
C
      M1=(IWST-1)*LW+1
      M2=IBUF-(10+LW+NFW+NIW+NAW+(5+LW+NAW)/2)+M1-1
C1
      NL=NW*LW
      N1=(IWST-1)*LW+1
      N2=N1+NL-1
      IF(M2.GT.N2)M2=N2
C
      WRITE(IU'IREC,ERR=97)LW,NW,NFW,NIW,NAW,NBW,NMBW,NMFW,
     *PNW,RNW,(IPW(I),I=1,LW),(FDOCW(J),J=1,NFW),
     *(IDOCW(K),K=1,NIW),(ADOCW(L),L=1,NAW),
     *(VW(M),M=M1,M2)
C1
       IF(JFLG.NE.1) GO TO 70
        IWORDS=(10+LW+NFW+NIW+NAW+(5+LW+NAW)/2)+(LW*NW)
C       IREC IS THE RECORD NO. FOR THE LAST RECORD IN THIS SEGMENT
        IREC=((IWORDS-1)/IBUF)+IREC
      GO TO 83
 70      IF(M2.EQ.N2)GO TO 83
      M1=M2+1
      M2=M2+IBUF
      IF(M2.GT.N2)M2=N2
      IREC=IREC+1
      WRITE(IU'IREC,ERR=97)(VW(J),J=M1,M2)
      GO TO 70
C
 82      M1=(IRST-1)*LR+1
      M2=IBUF-(10+LR+NFR+NIR+NAR+(5+LR+NAR)/2)+M1-1
C
      NL=NR*LR
      N1=(IRST-1)*LR+1
      N2=N1+NL-1
      IF(M2.GT.N2)M2=N2
C
      WRITE(IU'IREC,ERR=97)LR,NR,NFR,NIR,NAR,NBW,NMBR,NMFR,
     *PNR,RNR,(IPR(I),I=1,LR),(FDOCR(J),J=1,NFR),
     *(IDOCR(K),K=1,NIR),(ADOCR(L),L=1,NAR),
     *(VR(M),M=M1,M2)
C1
      IF(JFLG.NE.1) GO TO 71
        IWORDS=(10+LR+NFR+NIR+NAR+(5+LR+NAR)/2)+(LR*NR)
C       IREC IS THE RECORD NO. FOR THE LAST RECORD IN THIS SEGMENT
```

```
        IREC=((IWORDS-1)/IBUF)+IREC
      GO TO 83
 71     IF(M2.EQ.N2)GO TO 83
      M1=M2+1
      M2=M2+IBUF
      IF(M2.GT.N2)M2=N2
      IREC=IREC+1
      WRITE(IU'IREC,ERR=97)(VR(J),J=M1,M2)
      GO TO 71
C
   83 CONTINUE
C
C
      IPOS=IPOS+1
      ISECR(IU)=IREC+1
      IUNIT(IU)=IPOS
      JUNIT(IU)=IPOS
      GO TO (84,85),IW
   84 IF(B210) WRITE(6,1000)IU,NMFW,NBW,NMBW,PNW,RNW,NW,LW,NFW,NIW,NAW
 1000 FORMAT(' WRITE UNIT',I3,';  FILE ',A6,
     *   ';  SEGNUM',I4,';  SEGNAM ',A6,';  PN=',A6,';  RN=',A6,
     *  ';  N=',I6,';  L=',I4,'  NF=',I4,'  NI=',I4,'  NA=',I4)
C
C
      IF(B1) WRITE(6,1011) IU,NMFW,NBW,NMBW,PNW,RNW,NW,LW,NFW,NIW,NAW
 1011 FORMAT(' WRT ',I4,2X,A6,2X,I4,2X,3(A6,2X),I6,4I4)
C
      IF(B35) WRITE(6,1012)(IPW(I),I=1,LW)
 1012 FORMAT(' PARAMETERS: '12(2X,A6)/(13X,12(2X,A6)))
C
      IF(.NOT.B45) GO TO 110
      IF((NFW+NIW+NAW).EQ.0) GO TO 110
      WRITE(6,1013)
 1013 FORMAT(' ADDL DATA:')
      IF(NFW.GT.0)WRITE(6,1100)(FDOCW(I),I=1,NFW)
      IF(NIW.GT.0)WRITE(6,1101)(IDOCW(I),I=1,NIW)
      IF(NAW.GT.0)WRITE(6,1102)(ADOCW(I),I=1,NAW)
 1100 FORMAT(10(G11.5))
 1101 FORMAT(1X,12I6)
 1102 FORMAT(1X,12A6)
C
  110 IF(.NOT.B69) GO TO 107
      JL=IWST*LW
      J1=JL-LW+1
      WRITE(6,1014)(VW(I),I=J1,JL)
      JL=(NW+IWST-1)*LW
      J1=JL+1-LW
      WRITE(6,1015)(VW(J),J=J1,JL)
 1014 FORMAT(' FIRST CYCLE:',10(G11.5)/(13X,10(G11.5)))
 1015 FORMAT(' LAST CYCLE: ',10(G11.5)/(13X,10(G11.5)))
C
  107 IF(.NOT.B10) GO TO 108
      WRITE(6,1017)
      IQ1=IWST
```

```
      IQ2=IQ1+NW-1
      DO 106 I=IQ1,IQ2
      JL=I*LW
      J1=JL+1-LW
      WRITE(6,1016) I,(VW(J),J=J1,JL)
  106 CONTINUE
 1016 FORMAT(5X,I5,3X,10G12.6)
 1017 FORMAT(//' LISTING OF DATA'///)
C
      GO TO 86
   85 IF(B210) WRITE(6,1000)IU,NMFR,NBW,NMBR,PNR,RNR,NR,LR,NFR,NIR,NAR
C
      IF(B1) WRITE(6,1011) IU,NMFR,NBW,NMBR,PNR,RNR,NR,LR,NFR,NIR,NAR
C
      IF(B35) WRITE(6,1012)(IPR(I),I=1,LR)
C
      IF(.NOT.B45) GO TO 109
      IF((NFR+NIR+NAR).EQ.0) GO TO 109
      WRITE(6,1013)
      IF(NFR.GT.0)WRITE(6,1100)(FDOCR(I),I=1,NFR)
      IF(NIR.GT.0)WRITE(6,1101)(IDOCR(I),I=1,NIR)
      IF(NAR.GT.0)WRITE(6,1102)(ADOCR(I),I=1,NAR)
C
  109 IF(.NOT.B69) GO TO 117
      JL=IRST*LR
      J1=JL+1-LR
      WRITE(6,1014)(VR(I),I=J1,JL)
      JL=(NR+IRST-1)*LR
      J1=JL+1-LR
      WRITE(6,1015)(VR(J),J=J1,JL)
C
  117 IF(.NOT.B10) GO TO 108
      WRITE(6,1017)
      IQ1=IRST
      IQ2=IQ1+NR-1
      DO 116 I=IQ1,IQ2
      JL=I*LR
      J1=JL+1-LR
      WRITE(6,1016) I,(VR(J),J=J1,JL)
  116 CONTINUE
C
C
   86 CONTINUE
  108 IF=0
      IUP=IU
      RETURN
C
C
   95 IF=5
      WRITE(6,1005)NNNW,NNIP,NNF,NNI,NNA,
     *   NW,LW,NFW,NIW,NAW
 1005 FORMAT(//' A DIMENSION IS TOO SMALL.'//
     *   ' NNNW=',I6,'   NNIP=',I6,'   NNF=',I6,
     *   '   NNI=',I6,'   NNA=',I6//' NW=',I6,
```

```
     *   '   LW=',I6,'   NFW=',I6,'   NIW=',I6,'   NAW=',I6//)
      RETURN
   97 IF=3
      WRITE(6,1003) IU
 1003 FORMAT(' WRITE ERROR ON UNIT ',I3)
      LU=IU
      GO TO 90
   99 IF=1
      WRITE(6,1001) IU
 1001 FORMAT(' EOF ON UNIT ',I3)
      LU=IU
      GO TO 90
C
      ENTRY RESETW(KU)
      LU=KU
      CLOSE(UNIT=LU)
      IUSAV=0
 90      IREC=1
      IPOS=0
      IUNIT(LU)=0
      JUNIT(LU)=0
      ISECR(LU)=0
      RETURN
 9090      WRITE(6,*) ' ERROR IN OPENING UNIT ',IU
      END
```

Appendix B: A program for completing FEB file headers

A PROGRAM FOR COMPLETING FEB FILE HEADERS

The purpose of the program HEDFIL is to fill the header common areas of FEB files. This program is written entirely in FORTRAN 77. FEB utility subroutines ZREAD and ZWRIT are required. A brief description follows. The input FEB file is assigned to logical unit 3. All updates to the FEB file headers are made with direct-access writes to this file. It is a good idea to have a backup copy of the input file in case of user error while using this program. Input to the program is given by instruction file which consists of a one-letter key ($, F, I, A, S, or E) followed by the associated input. Instruction file input is described below.

$ Text string	Comment statement used for documentation, no effect on the processing.
F J,K,(FDOC(I),I=J,K)	FDOC words from word no. J through word no. K.
I J,K,(IDOC(I),I=J,K)	IDOC words from word no. J through word no. K.
A J,K,(ADOC(I),I=J,K)	ADOC words from word no. J through word no. K. A maximum of 12 six-character words per line may be input.
S ISEG	Reads the header for segment no. ISEG into core and writes all headers through segment no. ISEG-1.
E	Remainder of segment headers are written.

Note: No additional header words can be added with this program. All changes remain in effect through the end of the data set (i.e., IDOC(1)=1).

Sample UNIVAC Run:

```
@MAP,I A
IN .HEDFIL
IN .ZREAD
IN .ZWRIT
@END
@ASG,A FEBFILE.
@USE 3.,FEBFILE.
@XQT A
$ INSERT FDOC(1) - FDOC(3)
F 1,3,11.,22.,33.
$ INSERT IDOC(2) - IDOC(4)
I 2,4,10,20,30
```

```
$ INSERT FDOC(8)
F 8,8,44.3
$ INSERT ADOC(1) - ADOC(2)
A 1,2,TEST CASE 12
$ INSERT ADOC(13) - ADOC(16)
A 13,24,THIS IS A TEST LINE.
$ INSERT IDOC(9) - IDOC(10)
I 9,10,100,200
$ GET FIRST SEGMENT OF NEXT DATA SERIES
S 21
$ INSERT FDOC(1)
F 1,1,44.
$ WRITE REST OF FEB FILE
E
```

PROGRAM LISTING

```
C*******************************************************************************
C
C   PROGRAM:  HEDFIL
C   THIS PROGRAM FILLS THE FEB COMMON AREAS
C
C*******************************************************************************
C
        IMPLICIT REAL*4 (A-H,O-Z)
        IMPLICIT INTEGER*4 (I,J,K,L,M,N)
C
        CHARACTER*4 A
        CHARACTER*80 IN
        CHARACTER*6 ADOCR,IPR,NMFR,NMBR,PNR,RNR,AD
C
C   ZREAD COMMON AREAS
        COMMON /RHDR /LR,NR,NBR,NFR,NIR,NAR
        COMMON /RHDR1 /NMBR,NMFR,PNR,RNR,IPR(4)
        COMMON /RDOCF/FDOCR(500) /RDOCI/IDOCR(500) /RDOCA/ADOCR(500)
        COMMON / RDATA / VR(20000)
C
        COMMON / DIAGS / MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
C
        COMMON /DRDCOM/ JFLG
C
       DIMENSION FD(2,500),ID(2,500),AD(500),ADF(500)
C
       DATA NNNR,NNNW,NNIP,NNF,NNI,NNA /1000,1000,4,500,500,500/
       DATA IU,IOU,ISEG,K /3,-3,1,0/
C   SET FLAG FOR DIRECT ACCESS HEADER WRITE ONLY
       JFLG=1
C
C*******************************************************************
       WRITE(6,*)'HEDFIL: A FEB HEADER FILL PROGRAM'
C*******************************************************************
C
C   READ SEGMENT 1 INTO CORE
       CALL ZREAD(3,IF,1)
       IF(IF.GE.1)THEN
        WRITE(6,*)'PROBLEM WITH INPUT FILE'
        STOP
       END IF
C
       WRITE(6,*)'ENTER INSTRUCTIONS'
C
C*******************************************************************************
C
C   OPEN SCRATCH FILE TO ALLOW LIST DIRECTED READ
       CLOSE(UNIT=29)
       OPEN(UNIT=29,STATUS='SCRATCH',ERR=9096)
C
C   READ INPUT INTO INTERNAL FILE
 10    READ(5,5000,ERR=9000,END=999)IN
```

```
 5000  FORMAT(A)
C    WRITE INSTRUCTION TO UNIT 29
        WRITE(29,5000)IN(3:74)
        BACKSPACE 29
C
C*******************************************************************************
C
C    TRANSLATE INSTRUCTION
        A=IN(1:1)
C
C*******************************************************************************
C
C    READ COMMENT
        IF(A(1:1).EQ.'$'.OR.A(1:1).EQ.'$')GO TO 10
C
C*******************************************************************************
C
C    SET UP INSTRUCTIONS TO COMPLETE WRITING OF FEB FILE
        GO TO 20
 999    IN(3:9)='999999 '
        WRITE(29,5000)IN(3:9)
        BACKSPACE 29
        A='S'
  20    CONTINUE
C
C*******************************************************************************
C
C    READ SEGMENT NO. AND WRITE SEGMENTS UP TO SEGMENT NO.
        IF(A.EQ.'S'.OR.A.EQ.'s')THEN
         READ(29,*,ERR=9097)ISEG
C    IF ISEG IS 1, THIS SEGMENT HAS ALREADY BEEN READ
         IF(ISEG.EQ.1)GO TO 10
         K=K+1
C
         IF(K.GT.ISEG)THEN
          WRITE(6,*)'SEGMENT MUST BE GREATER THAN LAST SEGMENT READ'
          STOP
         END IF
C
         DO 40 I=K,ISEG-1
C
C    INCORPORATE CHANGES
         IF(JFD.EQ.1)THEN
          DO 32 J=1,NFR
 32        IF(FD(2,J).GT.0.5)FDOCR(J)=FD(1,J)
         END IF
C
         IF(JID.EQ.1)THEN
          DO 34 J=1,NIR
 34        IF(ID(2,J).EQ.1)IDOCR(J)=ID(1,J)
         END IF
C
         IF(JAD.EQ.1)THEN
          DO 36 J=1,NAR
```

```
 36      IF(ADF(J).GT.0.5)ADOCR(J)=AD(J)
        END IF
C
        ISW=NBR
        CALL ZWRIT(IOU,IF,ISW)
        IF(IF.GT.1)STOP
C   RESET FLAGS AT END OF SERIES
        IF(IDOCR(1).EQ.1)THEN
        DO 37 J=1,NFR
 37     FD(2,J)=0.
        DO 38 J=1,NIR
 38     ID(2,J)=0
        DO 39 J=1,NAR
 39     ADF(J)=0.
        JFD=0
        JID=0
        JAD=0
        END IF
C
        CALL ZREAD(IU,IF,0)
        IF(IF.EQ.1)THEN
         WRITE(6,*)'EOF REACHED IN INPUT FILE'
         STOP
        END IF
C
 40     CONTINUE
C
        K=ISEG
        GO TO 10
       END IF
C
C*************************************************************************
C
C   WRITE REST OF FEB FILE
       IF(A.EQ.'E'.OR.A.EQ.'e')GO TO 999
C
C*************************************************************************
C
C   READ FDOC INSTRUCTION
       IF(A.EQ.'F'.OR.A.EQ.'f')THEN
        READ(29,*,ERR=9097)I1,I2
C
        IF(I1.LE.0.OR.I1.GT.I2)THEN
         WRITE(6,*)'INDEX IS LESS THAN 0 OR 1ST INDEX IS GREATER ',
     *   'THAN 2ND'
         STOP
        END IF
C
        IF(I2.GT.NFR)THEN
         WRITE(6,*)'INDEX IS GREATER THAN MAX INDEX ALLOWED, NFR=',NFR
         STOP
        END IF
C
        BACKSPACE 29
```

```
        READ(29,*,ERR=9097)I1,I2,(FD(1,I),I=I1,I2)
        DO 42 I=I1,I2
 42     FD(2,I)=1
        JFD=1
        GO TO 10
       END IF
C
C*************************************************************************
C
C   READ IDOC INSTRUCTION
       IF(A.EQ.'I'.OR.A.EQ.'i')THEN
        READ(29,*,ERR=9097)I1,I2
C
        IF(I1.LE.0.OR.I1.GT.I2)THEN
         WRITE(6,*)'INDEX IS LESS THAN 0 OR 1ST INDEX IS GREATER ',
     *   'THAN 2ND'
         STOP
        END IF
C
        IF(I2.GT.NIR)THEN
         WRITE(6,*)'INDEX IS GREATER THAN MAX INDEX ALLOWED, NIR=',NIR
         STOP
        END IF
C
        BACKSPACE 29
        READ(29,*,ERR=9097)I1,I2,(ID(1,I),I=I1,I2)
        DO 44 I=I1,I2
 44     ID(2,I)=1
        JID=1
        GO TO 10
       END IF
C
C*************************************************************************
C
C   READ ADOC INSTRUCTION
       IF(A.EQ.'A'.OR.A.EQ.'a')THEN
        READ(29,*,ERR=9097)I1,I2
C
        IF(I1.LE.0.OR.I1.GT.I2)THEN
         WRITE(6,*)'INDEX IS LESS THAN 0 OR 1ST INDEX IS GREATER ',
     *   'THAN 2ND'
         STOP
        END IF
C
        IF(I2.GT.NAR)THEN
         WRITE(6,*)'INDEX IS GREATER THAN MAX INDEX ALLOWED, NAR=',NAR
         STOP
        END IF
C
C   THERE CAN BE NO MORE THAN 12 ADOC WORDS PER LINE
        IF(I2-I1+1.GT.12)THEN
         WRITE(6,*)'TOO MANY ADOC WORDS - ONLY 12/LINE ALLOWED'
         WRITE(6,*)'TRY AGAIN'
         GO TO 10
```

```
        END IF
C
C   FIND START OF ADOC TEXT
        ICOM=0
        DO 50 I=1,12
         ICHAR=I
         IF(IN(I:I).EQ.',')ICOM=ICOM+1
         IF(ICOM.EQ.2)GO TO 51
 50     CONTINUE
 51     CONTINUE
C
        IC1=ICHAR+1
        IC2=(I2-I1+1)*6+ICHAR
C
        JW=I1
        DO 60 I=IC1,IC2,6
        AD(JW)=IN(I:I+5)
 60     JW=JW+1
C
        DO 46 I=I1,I2
 46     ADF(I)=1
        JAD=1
        GO TO 10
       END IF
C
C*******************************************************************************
C
 9000  WRITE(6,*)'ERROR IN READING INSTRUCTION, TRY AGAIN'
       GO TO 10
 9096  WRITE(6,*)'ERROR IN OPENING UNIT 29, DO NOT USE UNIT 29!'
       STOP
 9097  WRITE(6,*)'ERROR WHILE READING INSTRUCTION ',A,' TRY AGAIN'
       GO TO 10
       END
```

```
      SUBROUTINE ZREAD(IU,IF,IBL)
C
C  THIS SUBROUTINE IS THE READ HALF OF AN INPUT-OUTPUT
C  PACKAGE FOR HANDLING NON-FORMATTED, ASCII FORTRAN
C  WRITTEN DATA FILES, COMMONLY REFERRED TO AS
C  FEB (FAST & EASY BINARY) FILES.
C
C
        IMPLICIT REAL*4 (A-H,O-Z)
        IMPLICIT INTEGER*4 (I,J,K,L,M,N)
C
      CHARACTER*6 ADOCR,IPR,NMFR,NMBR,PNR,RNR
      COMMON /RHDR /LR,NR,NBR,NFR,NIR,NAR
      COMMON /RHDR1 /NMBR,NMFR,PNR,RNR,IPR(1)
C      NOTE:  ORIGINAL ZREAD CONSISTED OF RHDR COMMON ALONE,
C      ASCII FORTRAN REQUIRED PLACING CHARACTER VARIABLES IN
C      SEPARATE COMMON - RHDR1.
      COMMON /RDOCF/FDOCR(1) /RDOCI/IDOCR(1) /RDOCA/ADOCR(1)
      COMMON / RDATA / VR(1)
C
      COMMON / DIAGS / MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
      COMMON / JPOS / JUNIT(99)
      COMMON/DRDCOM/ JFLG,ISECR(99)
      DIMENSION IUNIT(99)
      LOGICAL B1,B210,B10,B35,B45,B69,OD
      DATA MSGR / 2 /
      DATA LLSW/1/, IRST/1/
      DATA IBUF/600/
C
      B1=MSGR.EQ.1
      B210=MSGR.GE.2.AND.MSGR.LE.10
      B10=MSGR.EQ.10
      B35=MSGR.EQ.3.OR.MSGR.EQ.5.OR.MSGR.EQ.7.OR.MSGR.EQ.9.OR.MSGR.EQ.10
      B45=MSGR.EQ.4.OR.MSGR.EQ.5.OR.MSGR.GE.8.AND.MSGR.LE.10
      B69=MSGR.GE.6.AND.MSGR.LE.9
C
C
C      OPEN DIRECT ACCESS FEB FILES AS REQUIRED.
C      RECORD SIZE IS SET TO IBUF IN DATA STATEMENT.
C      EACH HOLLERITH WORD CONSISTS OF SIX CHARACTERS,
C      AND THUS OCCUPIES ONE AND ONE-HALF WORDS.
      IF(IU.NE.IUSAV)THEN
       INQUIRE(UNIT=IU,OPENED=OD)
       IF(.NOT.OD) OPEN(UNIT=IU,ACCESS='DIRECT',FORM='UNFORMATTED',
     *   STATUS='UNKNOWN',ERR=9090,RECL=IBUF)
       IUSAV=IU
      END IF
C
C
      IBLK=IBL
      IPOS=JUNIT(IU)
      IREC=ISECR(IU)
      IF(IPOS.EQ.0) IPOS=1
      IF(IREC.EQ.0)IREC=1
```

```
      IF(IBL.EQ.0) IBLK=IUNIT(IU)
      IF(IBLK.LT.IPOS) GO TO 5
    4 IF(IBLK.EQ.IPOS) GO TO 3
C
C       FULL DUMMY READ IS REQUIRED IN ORDER TO VERIFY RECORD.
C       FILES ARE ZERO-FILLED IN INITIAIZATION ON THE UNIVAC,
C       BUT ARE NOT ON THE VAX - ERR=99 BRANCH IS USED.
      READ(IU'IREC,ERR=99)LQ,NQ,NFQ,NIQ,NAQ,NBQ,(NMBQ,I=1,LQ+2),
     *(FDOCQ,I=1,NFQ),(IDOCQ,I=1,NIQ),(NMBQ,I=1,NAQ),
     *(VQ,M=((IRST-1)*LQ+1),(IRST-1)*LQ+IBUF
     *-(10+LQ+NFQ+NIQ+NAQ+(5+LQ+NAQ)/2))
      IF(LQ.EQ.0.OR.NBQ.EQ.0.OR.NFQ.LT.0.OR.NIQ.LT.0.OR.NAQ.LT.0)
     *GO TO 99
C       IWORDS IS THE TOTAL WORDS CONTAINED IN THE SEGMENT
      IWORDS=(10+LQ+NFQ+NIQ+NAQ+(5+LQ+NAQ)/2)+(LQ*NQ)
C       IREC IS THE RECORD NO. FOR THE NEXT SEGMENT
      IREC=((IWORDS-1)/IBUF)+1+IREC
      IPOS=IPOS+1
      IUNIT(IU)=IPOS
      JUNIT(IU)=IPOS
      GO TO 4
C
    5 IF=0
      IREC=1
      IF(IBL.EQ.0) IBLK=1
      IPOS=1
      IUNIT(IU)=IPOS
      JUNIT(IU)=IPOS
      ISECR(IU)=0
      GO TO 4
    3 CONTINUE
C
C     THIS DUMMY READ IS NECESSARY FOR END OF FILE CHECKING,
C     OTHERWISE, ARRAY LIMITS CAN EASILY BE EXCEEDED.
      READ(IU'IREC,ERR=99)LQ,NQ,NFQ,NIQ,NAQ,NBQ,(NMBQ,I=1,LQ+2),
     *(FDOCQ,I=1,NFQ),(IDOCQ,I=1,NIQ),(NMBQ,I=1,NAQ),
     *(VQ,M=((IRST-1)*LQ+1),(IRST-1)*LQ+IBUF
     *-(10+LQ+NFQ+NIQ+NAQ+(5+LQ+NAQ)/2))
      IF(LQ.EQ.0.OR.NBQ.EQ.0.OR.NFQ.LT.0.OR.NIQ.LT.0.OR.NAQ.LT.0)
     *GO TO 99
C
      M1=(IRST-1)*LQ+1
      M2=(IRST-1)*LQ+IBUF-(10+LQ+NFQ+NIQ+NAQ+(5+LQ+NAQ)/2)
C1
      NL=NQ*LQ
      N2=M1+NL-1
      IF(M2.GT.N2)M2=N2
C
C
      READ(IU'IREC,ERR=99)LR,NR,NFR,NIR,NAR,NBR,NMBR,NMFR,PNR,RNR,
     *(IPR(I),I=1,LR),(FDOCR(J),J=1,NFR),
     *(IDOCR(K),K=1,NIR),(ADOCR(L),L=1,NAR),
     *(VR(M),M=M1,M2)
      IF(NR.GT.NNNR.OR.LR.GT.NNIP.OR.NFR.GT.NNF.
```

```
     *   OR.NIR.GT.NNI.OR.NAR.GT.NNA) GO TO 95
C1
      IF(JFLG.NE.1) GO TO 8
      IWORDS=(10+LR+NFR+NIR+NAR+(5+LR+NAR)/2)+(LR*NR)
C      IREC IS THE RECORD NO. FOR THE LAST RECORD IN THIS SEGMENT
      IREC=((IWORDS-1)/IBUF)+IREC
      GO TO 9
 8      IF(M2.EQ.N2)GO TO 9
      M1=M2+1
      M2=M2+IBUF
      IF(M2.GT.N2)M2=N2
      IREC=IREC+1
      READ(IU'IREC,ERR=99)(VR(J),J=M1,M2)
      GO TO 8
    9 CONTINUE
C
C
      IPOS=IPOS+1
      IUNIT(IU)=IPOS
      JUNIT(IU)=IPOS
      ISECR(IU)=IREC+1
C
      IF(MSGR.EQ.0) GO TO 108
      IF(B210) WRITE(6,1000)IU,NMFR,NBR,NMBR,PNR,RNR,NR,LR,NFR,NIR,NAR
 1000 FORMAT(' READ UNIT',I3,';  FILE ',A6,
     *   ';  SEGNUM',I4,';  SEGNAM ',A6,';  PN=',A6,';  RN=',A6,
     *   ';  N=',I6,';  L=',I4,'  NF=',I4,'  NI=',I4,'  NA=',I4)
C
      IF(B1) WRITE(6,1011) IU,NMFR,NBR,NMBR,PNR,RNR,NR,LR,NFR,NIR,NAR
 1011 FORMAT(' RD ',I4,2X,A6,2X,I4,2X,3(A6,2X),I6,4I4)
C
      IF(B35) WRITE(6,1012)(IPR(I),I=1,LR)
 1012 FORMAT(' PARAMETERS: '12(2X,A6)/(13X,12(2X,A6)))
C
      IF(.NOT.B45) GO TO 110
      IF((NFR+NIR+NAR).EQ.0) GO TO 110
      WRITE(6,1013)
 1013 FORMAT(' ADDL DATA:')
      IF(NFR.GT.0)WRITE(6,1100)(FDOCR(I),I=1,NFR)
      IF(NIR.GT.0)WRITE(6,1101)(IDOCR(I),I=1,NIR)
      IF(NAR.GT.0)WRITE(6,1102)(ADOCR(I),I=1,NAR)
 1100 FORMAT(10G11.5)
 1101 FORMAT(1X,12I6)
 1102 FORMAT(1X,12A6)
C
  110 IF(.NOT.B69) GO TO 107
      JL=IRST*LR
      J1=JL-LR+1
      WRITE(6,1014)(VR(I),I=J1,JL)
      JL=(NR+IRST-1)*LR
      J1=JL+1-LR
      WRITE(6,1015)(VR(J),J=J1,JL)
 1014 FORMAT(' FIRST CYCLE:',10G11.5/(13X,10G11.5))
 1015 FORMAT(' LAST CYCLE: ',10G11.5/(13X,10G11.5))
```

```
C
  107 IF(.NOT.B10) GO TO 108
      WRITE(6,1017)
      IQ1=IRST
      IQ2=IQ1+NR-1
      DO 106 I=IQ1,IQ2
      JL=I*LR
      J1=JL+1-LR
      WRITE(6,1016) I,(VR(J),J=J1,JL)
  106 CONTINUE
 1016 FORMAT(5X,I5,3X,10G12.6)
 1017 FORMAT(//' LISTING OF DATA'///)
C
  108 IF=0
      IUP=IU
      RETURN
C
C
   95 IF=5
      WRITE(6,1005)NNNR,NNIP,NNF,NNI,NNA,
     *   NR,LR,NFR,NIR,NAR
 1005 FORMAT(//' A DIMENSION IS TOO SMALL.'//
     *   ' NNNR=',I6,'   NNIP=',I6,'   NNF=',I6,
     *   '   NNI=',I6,'   NNA=',I6//' NR=',I6,
     *   '   LR=',I6,'   NFR=',I6,'   NIR=',I6,'   NAR=',I6//)
      RETURN
   99 IF=1
      WRITE(6,1001) IU
 1001 FORMAT(' EOF ON UNIT ',I3)
 90       IREC=1
      IPOS=0
      IUNIT(IU)=IPOS
      JUNIT(IU)=IPOS
      ISECR(IU)=0
      RETURN
 9090      WRITE(6,*) ' ERROR IN OPENING UNIT ',IU
      END
```

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A167083

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NORDA Report 137	5. MONITORING ORGANIZATION REPORT NUMBER(S) NORDA Report 137
6. NAME OF PERFORMING ORGANIZATION Naval Ocean Research and Development Activity	7a. NAME OF MONITORING ORGANIZATION Naval Ocean Research and Development Activity
6c. ADDRESS (City, State, and ZIP Code) Ocean Science Directorate NSTL, Mississippi 39529–5004	7b. ADDRESS (City, State, and ZIP Code) Ocean Science Directorate NSTL, Mississippi 39529–5004
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Naval Oceanographic Office 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER Work Request 50045
8c. ADDRESS (City, State, and ZIP Code) NSTL, Mississippi 39529–5004	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.

11. TITLE (Include Security Classification)
A FORTRAN 77 Computer Program for Processing Raw CTD Data to ASCII FEB File Format

12. PERSONAL AUTHOR(S)
William J. Teague

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	From _____ To _____	February 1986	49

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	FORTRAN, ASCII, FEB, CTD

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

A software package written in Fortran 77 has been developed for processing raw conductivity, temperature, and depth (CTD) data to ASCII Fast and Easy Binary (FEB) format. The programs contained within the package are machine independent with the exception of the tape block-read routine. Briefly, the tape header information is decoded and checked against information input by the program user, the raw data are read from the data-logger tape and converted to engineering units, the data are screened for spurious data values and, finally, the data are written to FEB files. Testing was performed on VAX 11/750 and UNIVAC 1180 machines.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ■ DTIC USERS ☐	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL William J. Teague	22b. TELEPHONE NUMBER (Include Area Code) (601) 688-4734	22c. OFFICE SYMBOL Code 331

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

END

FILMED

5-86

DTIC