NOAA, NATIONAL WEATHER SERVICE, CENTRAL REGION

COMPUTER PROGRAMS AND PROBLEMS

NWS CRCP - NO. 11

METEOROLOGICAL APPLICATIONS LIBRARY (CRH.LB)

Thomas F. Schwein
National Weather Service
Central Region Headquarters
Kansas City, MO

November 1983

U.S. DEPARTMENT OF COMMERCE / National Oceanic and Atmospheric Administration / National Weather Service

NOAA, National Weather Service
Central Region Computer Programs and Problems Series

NWS CRCP No. 1 A Computer Program for Determining and Displaying the Geostrophic Winds on Lakes Superior and Michigan. R. Somrek. June 1980 (PB80 225675)

NWS CRCP No. 2 Program ALEMBIC. W. Sunkel. March 1982 (PB83 114488)

NWS CRCP No. 3 A Subroutine to Find and Extract Data from an AFOS Database Product. R. Somrek. May 1982 (PB83 118828)

NWS CRCP No. 4 Programs SAVALL and STORALL. W. Sunkel. July 1982. (PB83 118448)

NWS CRCP No. 5 Program ANALYZ. T. Schwein. Sept. 1982

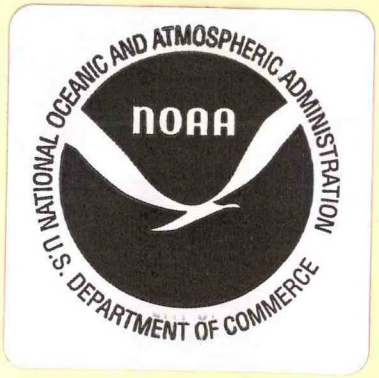
NWS CRCP No. 6 A Regional Weather Depiction Plot. W. Sunkel. October 1982. (PB83 194415)

NWS CRCP No. 7 The Topeka Library (TOP.LB). W. Sunkel. March 1983

NWS CRCP No. 8 A Multipurpose Weather Roundup Program. Warren E. Sunkel May 1983. (PB83 222612)

NWS CRCP No. 9 Assembly Language Graphics Library (EGR1.LB) with Fortran Interfacing. Thomas J. Egger July 1983 (PB83 239624)

NWS CRCP No. 10 A Thermodynamics Library for AFOS Programmers (THERMO.LB) Warren E. Sunkel, November 1983



NOAA, NATIONAL WEATHER SERVICE, CENTRAL REGION
Computer Programs and Problems

NWS CRCP - No. 11

METEOROLOGICAL APPLICATIONS LIBRARY (CRH.LB)

Thomas F. Schwein
National Weather Service
Central Region Headquarters
Kansas City, MO

November 1983

UNITED STATES
DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Oceanic and
Atmospheric Administration
John V. Byrne, Administrator

National Weather
Service
Richard E. Hallgren,
Assistant Administrator

METEOROLOGICAL APPLICATIONS LIBRARY

(CRH.LB)

Thomas F Schwein
National Weather Service Office
Central Region Headquarters
Kansas City Missouri

INTRODUCTION

The Meteorological Applications Library is a set of subroutines that perform tasks in four distinct areas. Briefly, these are

1. METEOROLOGICAL CALCULATIONS:

   A). Dry Adiabatic Ascension
   B). Wet Adiabatic Ascension
   C). Calculation of Precipitable Water, Mixing Ratio and other Moisture Parameters.
   D). SWEAT Index Calculation
   E). Vapor Pressure Calculation
   F). Mean Wind Vector Calculation
   G). Pressure Level Height Calculation
   H). Calculate the Convective Condensation Level

2. DATABASE MANIPULATION

   A). Retrieve an AFOS Product in its entirity with WMO headers stripped.
   B). Search for Station Information in STDIR.MS

3. MATHEMATICAL CALCULATIONS

   A). Vertical Interpolation
   B). Conversion of Radians to Compass Degrees
   C). Conversion of Compass Degrees to Radians
   D). Two Dimensional Curve Fitting with Polynomials
   E). Interpolation with Polynomials
   F). Least Squares Matrix Solution

4. APPLICATION PROGRAM UTILITIES

   A). Convert ASCII to Floating Point
   B). Convert ASCII to Integer Binary
   C). Convert Floating Point to ASCII
   D). Convert Integer Binary to ASCII
   E). Find Previous or Next Day
   F). Find the Date/Time in Local Time Zone

I. IDENTIFICATION

Module name: SWEAT
Date: October 10 1983
Function: Calculate the SWEAT (Severe Weather Threat) Index
Language: Fortran

II. FUNCTIONAL SUMMARY

SWEAT calculates the SWEAT Index. If the surface pressure is below 850 MB, SWEAT will estimate 850MB conditions from the 700 and 500 MB reports. This makes the subroutine useful for stations in high mountainous country.

III. CALLING METHOD

CALL SWEAT(TMAN,DPMAN,IWD,IWS,ISWEAT,TS,DPS)

TMAN-Input floating point array of temperatures at the mandatory levels in degrees Celsius. Order of reports must be 1000, 850, 700,500,400,300,250,200,150,100 MB. Missing levels must be set to -99.0.

DPMAN-Input floating point array of dew points at the mandatory levels in degrees Celsius. Order of reports must be the same as for TMAN. Missing levels must be set to -99.0.

WD-Input integer array of wind directions at the mandatory levels, in whole degrees of the compass. Order of reports must be the same as for TMAN. Missing levels must be set to -99.

WS-Input integer array of wind speeds at the mandatory levels, in knots. Order of reports must be the same as for TMAN. Missing levels must be set to -99.

ISWEAT-Output SWEAT Index. If the SWEAT Index could not be calculated, the subroutine will return a value of -99 for ISWEAT.

TS-Input floating point temperature at the surface as reported in the mandatory level report, in degrees Celsius.

DPS-Input floating point dew point temperature at the surface as reported in the mandatory level report, in degrees Celsius.

These routines are all located in the library entitled CRH.LB.

LOADING

CRH.LB uses various subroutines located in BG.LB and UTIL.LB. Therefore, CRH.LB must be loaded before these libaries, so that any externals references created by CRH.LB will be satisfied by BG.LB and UTIL.LB.

```
      SUBROUTINE SWEAT(TMAN,DPMAN,WD,WS,ISWEAT,TS,DPS)
      DIMENSION TMAN(10),DPMAN(10)
      INTEGER WD(10),WS(10)
      ISWEAT=-99
      IF(TMAN(2).GT.-50.0) GO TO 1;SOUNDING ABOVE 850 MPH
3     T850=TS
      DP850=DPS
      WD850=2*WD(3)-WD(4);ESTIMATE OF 850 MB WIND
      IF(WD850.LT.0.0) WD850=WD850+360.0
      WS850=0.7*WS(3);ESTIMATE OF 850 MB WIND
      IF(T850.LT.-50.0.OR.DP850.EQ.-99.0.OR.WD850.LT.0.0.OR.
     1WS850.LT.0) RETURN
      GO TO 2
1     T850=TMAN(2)
      DP850=DPMAN(2)
      WD850=WD(2)
      WS850=WS(2)
      IF(DP850.EQ.-99.0.OR.WD850.LT.0.0.OR.WS850.LT.0.0) GO TO 3
 2    IF(TMAN(4).EQ.-99.0) RETURN
      TOTALS=T850+DP850-2*TMAN(4)
      IF(WD(4).LT.0) RETURN
      S1=WD(4)-WD850
      S=SIN(S1/57.3)
      D=DP850
      F8=WS850
      IF(WS(4).LT.0) RETURN
      F5=WS(4)
      S125=125*(S+0.2)
      IF(WD(4).LT.210.OR.WD(4).GT.310) S125=0.0
      IF(WD850.LT.130.0.OR.WD850.GT.250.0) S125=0.0
      TEST=WD(4)-WD850
      IF(TEST.LE.0.0) S125=0.0
      IF(WS(4).LT.15) S125=0.0
      IF(WS850.LT.15.0) S125=0.0
      SINDX=2*F8+F5+S125
      IF(TOTALS.GE.49.0) SINDX=SINDX+20*(TOTALS-49.0)
      IF(D.GE.0.0) SINDX=SINDX+12*D
      ISWEAT=SINDX+0.5
      RETURN
      END
```

I. IDENTIFICATION

Module name: MWIND
Date: October 10 1983
Function: Calculate the Mean Wind Vector from observed Rawinsonde data
Language: Fortran

II. FUNCTIONAL SUMMARY

MWIND calculates the Mean Wind Vector from the reported winds at up to 10 levels. It separates each reported wind into it's vector u and v components, and then calculates the vector average of all the reports.

III. CALLING METHOD

CALL MWIND(WD,WS,IWD,IWS)

WD-Floating point array, dimensioned to 10, containing observed wind directions. Unused elements of this array are set to -1 to indicate the absense of data.

WS-Floating point array, dimension to 10, containing observed wind speeds.

IWD-Calculated mean wind direction

IWS-Calculated mean wind speed

```
      SUBROUTINE MWIND(WD,WS,IWD,IWS)
      INTEGER WD(10),WS(10)
      ICOUNT=0
      V=0.0
      U=0.0
      DO 1 I=1,10
      IF(WD(I).LT.0.OR.WS(I).LT.0) GO TO 1
      ICOUNT=ICOUNT+1
      W=450-WD(I)
      IF(W.GT.360.0) W=W-360.0
      WX=WS(I)*COS(W/57.3)
      WY=WS(I)*SIN(W/57.3)
      U=U+WX
      V=V+WY
1     CONTINUE
      IWD=-99
      IWS=-99
      IF(ICOUNT.LE.5) RETURN
      U=U/ICOUNT
      V=V/ICOUNT
      SPEED=SQRT(U*U+V*V)
      IWS=SPEED+0.5
      DIR=ATAN2(V,U)
      DIR=DIR*57.3
      DIR=450-DIR
      IF(DIR.GT.360.0) DIR=DIR-360
      IF(DIR.GT.360.0) DIR=DIR-360
      IWD=DIR
      RETURN
      END
```

I. IDENTIFICATION

Module name: GTROF
Date: October 10 1983
Function: Calculate height in feet MSL of any pressure level from observed rawinsonde data
Language: Fortran

II. FUNCTIONAL SUMMARY

GTROF will calculate the height in feet MSL of any pressure level from observed rawinsonde data.

III. CALLING METHOD

CALL GTROF(IPRES,HEIGHT,H,IER)

IPRES-Input integer pressure level at which to calculate height. The pressure level chosen does not have to be at one of the mandatory or significant levels in the rawinsonde report.

HEIGHT-Output floating point calculated height in feet MSL of the pressure level IPRES

H-Floating point array of observed heights (in meters) from the rawinsonde run, ordered as follows:

H(1)-1000MB Height	H(6)-300MB Height
H(2)-850MB Height	H(7)-250MB Height
H(3)-700MB Height	H(8)-200MB Height
H(4)-500MB Height	H(9)-150MB Height
H(5)-400MB Height	H(10)-100MB Height

Missing data must be coded as -99.0

IER-Error return. If equal to zero, successful return. If non-zero, this indicates that the height could not be calculated.

```
      SUBROUTINE GTROP(TROP,HEIGHT,H,IER)
C FIND TROP HEIGHT IN FEET GIVEN HEIGHTS AT STANDARD LEVELS
      INTEGER TROP,P
      DIMENSION H(1)
      COMMON/PR/P(10)
      DATA P/1000,850,700,500,400,300,250,200,150,100/
C TROP...PRESSURE AT TROP
C HEIGHT..RETURNED HEIGHT AT TROP
C H...ARRAY OF HEIGHTS AT 100,850,700,500,400,300,250,200,150,100
C NUM...NUMBER OF VALUES IN ARRAY H...NORMALLY 10
C IER...ERROR CODE...0 MEAN ALL OK
C
C IT IS ASSUMED THAT HEIGHTS ARE IN CORRECT ORDER
C CHECK THAT TROP HEIGHT IS WITHIN BOUNDS OF GIVEN HEIGHTS
      DO 1 I=1,10
      IF(TROP.GE.P(I)) GO TO 2
 1    CONTINUE
      HEIGHT=-99
      IER=1
      RETURN
 2    CONTINUE
C FIT CURVE Y=MX+B...PRES IN P**.287
      X1=P(I-1)
      X2=P(I)
      IER=1
      IF(H(I-1).EQ.-99.OR.H(I).EQ.-99) RETURN
      Y1=H(I-1)
      Y2=H(I)
      X1=X1**0.287
      X2=X2**0.287
      X3=TROP
      X3=X3**0.287
      DYDX=(Y2-Y1)/(X2-X1)
      B=Y1-DYDX*X1
      Y3=DYDX*X3+B
      HEIGHT=Y3*3.2808399
      RETURN
      END
```

I. IDENTIFICATION

Module name: AVAP
Date: October 10 1983
Function: Calculate Temperature corresponding to a given saturation vapor pressure
Language: Fortran

II. FUNCTIONAL SUMMARY

Given a saturation vapor pressure, AVAP will return the Temperature in degrees Celsius corresponding to this vapor pressure.

III. CALLING METHOD

AVAP is a FORTRAN function.

A=AVAP(E)

E=Input saturation vapor pressure in millibars

Remarks: AVAP uses a simplified relation for vapor pressure that is valid at sea level and gives vapor pressure within plus or minus 1% at 100MB.

The temperature that is returned by AVAP is actually the Dew Point temperature, since the vapor pressure is assumed to be the saturation vapor pressure.

```
      REAL FUNCTION AVAP(VAP)
C FIND TEMPERATURE IN DEGREES C THAT CORRESPONDS TO VAPOR PRESSURE
      AVAP=237.3*(ALOG10(VAP/6.11))/(7.5-(ALOG10(VAP/6.11)))
      RETURN
      END
```

I. IDENTIFICATION

Module name: GVAP
Date: October 10 1983
Function: Calculate Saturation Vapor pressure
Language: Fortran

II. FUNCTIONAL SUMMARY

Given a temperature, GVAP will calculate the saturation vapor pressure in millibars.

III. CALLING METHOD

GVAP is a FORTRAN function.

A=GVAP(T)

T=Input temperature in degrees Celsius

Remarks: GVAP uses a simplified relation for vapor pressure that is valid at sea level and gives vapor pressure within plus or minus 1% at 100MB.

```
      REAL FUNCTION GVAP(T)
C FIND SATURATION VAPOR PRESSURE AT TEMP T...T IN DEGREES C
      GVAP=6.11*(10.0**(7.5*T/(237.3+T)))
      RETURN
      END
```

I. IDENTIFICATION

Module name:	DRY
Date:	October 10 1983
Function:	Given initial conditions at (Ti,Pi), calculate the temperature at PZ using an adiabatic process
Language:	Fortran

II. FUNCTIONAL SUMMARY

DRY calculates the temperature (in degrees Celsius) at pressure level PZ, given initial conditions at temperature Ti and pressure level Pi

III. CALLING METHOD

DRY is a FORTRAN Function

A=DRY(Ti,Pi,PZ)

Ti=Floating point temperature, in degrees Celsius, at pressure Pi

Pi-Floating point pressure, in millibars, initial pressure level

PZ-Floating point pressure, in millibars, final pressure level

```
      FUNCTION DRY(T1,P1,P2)
C GO ADIABATICALLY FROM T1,P1 TO T2,P2
      DRY=((T1+273.16)*(P2/P1)**0.287)-273.16
      RETURN
      END
```

I. IDENTIFICATION

Module name: PSEUDO
Date: October 10 1983
Function: Pseudo-adiabatic ascension (descension) routine
Language: Fortran

II. FUNCTIONAL SUMMARY

Given initial conditions at level (P1,T1), PSEUDO will calculate the new temperature T2 at input level P2 using a Pseudo-adiabatic process.

III. CALLING METHOD

PSEUDO is a FORTRAN function.

A=PSEUDO(P1,T1,P2)

P1=Initial pressure level in Millibars
T1=Initial temperature at level P1 in degrees Celsius
P2=Final pressure level in Millibars

```
      FUNCTION PSEUDO(P1,T1,P2)
      T1=T1+273.16
      TH0=THETAEQ(P1,T1,C,XL)
      A1=T1*(1.0+2.0*(P2-P1)*(.286+XL)/(P2+P1)/C)
      DO 100 I=1,2
      THE=THETAEQ(P2,A1,C,XL)
      A1=A1*(1.0+(TH0-THE)/THE/C)
  100 CONTINUE
      PSEUDO=A1-273.16
      T1=T1-273.16
      RETURN
      END
```

```
      FUNCTION THETAEQ(P,T,C,XL)
C COMPUTE EQUIVALENT POT TEMP
      T5=T-273.16
      E=GVAP(T5)
      PP=P-E
      W=0.62197*E/PP
      G=3146./T
      XL=(G-2.405)*W
      CHI=XL-0.601806+1363.7/(1348.53+PP-95289.7/(160.380+PP))
      THETAEQ=T*(5.71973/(2.36532-CHI)-1.41905)
      C=1+4226.*T*XL/(T-31.6)**2-G*W
      RETURN
      END
```

I. IDENTIFICATION

Module name: MRH
Date: October 10 1983
Function: Calculate various moisture parameters from sounding data
Language: Fortran

II. FUNCTIONAL SUMMARY

MRH calculates various moisture parameters from observed rawinsonde data. Parameters calculated are:

Precipitable Water-The precipitable water in the column in inches is calculated up to 100 MB. The calculation of precipitable water in this routine is done on the significant levels, and thus will vary slightly from the NMC derived value which uses only Mandatory levels.

Moisture Depth-Moisture Depth is depth in Millibars of Temperature-Dew point spreads less than or equal to 5 degrees Centigrade. It is calculated up to a specified level Li on a millibar by millibar basis.

Mean Mixing Ratio-The Mean mixing ratio in Grams per Kilogram is calculated over a mixing depth L2

Mean Relative Humidity-The mean relative humidity in the column is calculated over a mixing depth L2

III. CALLING METHOD

CALL MRH(P,T,DP,NUM,AMR,PW,IDEPTH,IRH,Li,L2)

P-Input floating point pressure level array in Millibars

T-Input floating point temperature array in degrees Celsius

DP-Input floating point Dew Point array in degrees Celsius

NUM-Input number of levels in P,T and DP (Integer)

AMR- Output Floating Point Mean Mixing Ratio in Grams per Kilogram over the Mixing depth L2.

PW-Output Floating Point Precipitable Water in inches from the surface to 100 Millibars. If the sounding ends below 100 MB, the routine returns the Precipitable Water up to the last level.

IDEPTH-Output Integer Moisture depth from the surface to the specified level Li.

IRH-Output Integer Mean Relative Humidity over the Mixing Depth L2

LI-Input Integer Pressure level in Millibars which to compute Moisture Depth up to.

LI-Input Integer Mixing depth in Millibars which to compute various parameters over

```
      SUBROUTINE MRH(P,T,DP,NUM,AMR5,PW,IDEPTH,IRH,IL,IMX)
C CALCULATE VARIOUS MOISTURE PARAMETERS FROM A SOUNDING
C
C P...ARRAY OF PRESSURES (INPUT)
C T...ARRAY OF TEMPS  (INPUT)
C DP...ARRAY OF DEW POINTS (INPUT)
C NUM...NUMBER OF LEVELS IN SOUNDING (INPUT)
C AMR5...AVG MIXING RATIO UP TO LEVEL L1  (OUTPUT)
C PW...PRECIPITABLE WATER IN INCHES (OUTPUT)
C IDEPTH...DEPTH OF TEMP-DP SPREAD LESS THAN 5 DEGREES FROM SFC IN MB (OUTPUT)
C IRH...AVG RH UP TO LEVEL L1 (OUTPUT)
C L1...LEVEL UP TO WHICH TO CALCULATE AVG MIXING RATIO/AVG HMDY
      DIMENSION P(1),T(1),DP(1)
      L1=IL
      IDEPTH=0
      PW=0.0
      PSTOP=L1
      AMR=0.0
      ICODE=1
      ASMR=0.0
      ICOUNT=0
      P7=P(1)-IMX;TOP LEVEL FOR COMPUTING MEAN MIXING RATIO
C FIT LINE Y=MX+B TO EVERY TWO LEVELS...Y=P X=DP Z=T
      DO 1 ILVLS=2,NUM
C
      SLPD=((P(ILVLS)**0.287)-(P(ILVLS-1)**0.287))/(DP(ILVLS)-DP(
     1ILVLS-1)+0.001)
      BD=(P(ILVLS)**0.287)-SLPD*DP(ILVLS)
      SLPT=((P(ILVLS)**0.287)-(P(ILVLS-1)**0.287))/(T(ILVLS)-T(
     1ILVLS-1)+0.001)
      BT=(P(ILVLS)**0.287)-SLPT*T(ILVLS)
C CALCULATE PRECIPITABLE WATER
      ADP=(DP(ILVLS)+DP(ILVLS-1))/2.0;MEAN DEW POINT IN LAYER
      ED=GVAP(ADP);MEAN VAPOR PRESSURE IN LAYER
      AMR1=0.621*ED/((P(ILVLS-1)+P(ILVLS))/2.0-ED)
      PW=PW+AMR1*(P(ILVLS-1)-P(ILVLS))
C GO UP FITTED LINE EVERY MB AND FIND MIXING RATIO
      START=P(ILVLS-1)+1
      DO 2 JLVLS=1,500
      START=START-1.0
      IF(START.LE.P(ILVLS)) GO TO 1;GO TO NEXT SET OF LEVELS
C CALCULATE DP
      Y=START**0.287
      DP1=(Y-BD)/SLPD
C CALCULATE T
      T1=(Y-BT)/SLPT
C CALCULATE MOISTURE DEPTH
      DEPTH=T1-DP1
      IF(DEPTH.LT.5.5.AND.START.GE.PSTOP) IDEPTH=IDEPTH+1
C FIND VAPOR PRES AND SATURATION VAPOR PRES
      ED=GVAP(DP1)
      ET=GVAP(T1)
C CALCULATE MIXING RATIO AND SATURATION MIXING RATIO
      AMR1=0.621*ED/(START-ED)
      AMR2=0.621*ET/(START-ED)
      IF(START.LT.P7) GO TO 1;ABOVE LEVEL TO CALCULATE MEAN MXG RATIO
      AMR=AMR+AMR1
      ASMR=ASMR+AMR2
      ICOUNT=ICOUNT+1
 2    CONTINUE
```

```
1     CONTINUE
      PW=PW*0.4
      AMR=AMR/ICOUNT
      ASMR=ASMR/ICOUNT
      AMR5=AMR*1000.0
      RH=AMR/ASMR
      IRH=RH*100+0.5
      RETURN
      END
```

I. IDENTIFICATION

Module name:	CCL
Date:	October 10 1983
Function:	Find the Convective Condensation Level
Language:	Fortran

II. FUNCTIONAL SUMMARY

CCL calculates the pressure level of the Convective Condensation level, given initial conditions of temperature, pressure and mixing ratio.

III. CALLING METHOD

CALL CCL(T,P,AMR,CL)

T-Input floating point temperature in degrees Celsius
P-Input floating point pressure in Millibars
AMR-Input floating point mixing ratio in grams per gram
CL-Output floating point Convective Condensation Level pressure

```
      SUBROUTINE CCL(T1,P1,AMX,CL)
C FIND CCL GIVEN INITIAL CONDITIONS T1,P1,AMX
      DRY(P0,P2,T)=((T+273.16)*((P2/P0)**0.286))-273.16
C CALCULATE INITIAL CONDITIONS
      E=GVAP(T1)
      AMXLO=(0.622*E)/(P1-E)
      IF(AMXLO.LE.AMX) CL=P1
      IF(AMXLO.LE.AMX) RETURN
C ITERATE TO NEAREST 100 MB
      DO 1 I=1,10
      PTEST=P1-I*100
      TTEST=DRY(P1,PTEST,T1)
      E=GVAP(TTEST)
      AMXLO=(0.622*E)/(PTEST-E)
      IF(AMXLO.EQ.AMX) CL=PTEST
      IF(AMXLO.EQ.AMX) RETURN
      IF(AMXLO.LT.AMX) GO TO 2;TOO HIGH
 1    CONTINUE
 2    PSTART=PTEST+100;GO BACK DOWN 100 MB
C ITERATE TO NEAREST 10 MB
      DO 3 I=1,10
      PTEST=PSTART-I*10
      TTEST=DRY(P1,PTEST,T1)
      E=GVAP(TTEST)
      AMXLO=(0.622*E)/(PTEST-E)
      IF(AMXLO.EQ.AMX) CL=PTEST
      IF(AMXLO.EQ.AMX) RETURN
      IF(AMXLO.LT.AMX) GO TO 5;TOO HIGH
 3    CONTINUE
 5    PSTART=PTEST+10
C ITERATE TO NEAREST MILLIBAR
      DO 6 I=1,10
      PTEST=PSTART-I
      TTEST=DRY(P1,PTEST,T1)
      E=GVAP(TTEST)
      AMXLO=(0.622*E)/(PTEST-E)
      IF(AMXLO.LE.AMX) GO TO 7
 6    CONTINUE
 7    CL=PTEST
      RETURN
      END
```

I. IDENTIFICATION

Module name: GETTX
Date: October 10 1983
Function: Database retrieval
Language: Fortran
Memory:

II. FUNCTIONAL SUMMARY

GETTX returns the latest version of an AFOS product in its entirity (up to 8 blocks long) with the AFOS comms header and WMO header stripped.

III. CALLING METHOD

CALL GETTX(ID,IBUF,NBYTES,IER)

ID-10 Byte Array contain PIL of product to retrieve
IBUF-Array to recieve packed text. Must be dimensioned large enough to hold largest product expected (up to 2048 bytes)
NBYTES-Returned number of bytes of product in IBUF
IER- Error return. If 0-indicates no product available or product longer than 8 blocks. If equal to 1, indicates a successful return.

```
      SUBROUTINE GETTX(ID,IBUF,NBYTES,IER)
C READ IN AFOS MESSAGE-STRIP OFF HEADER INFO
      INTEGER IBUF(1),ID(5),JBUF(128),KBUF(256),IHOLD(2048)
C RETRIEVE UP TO AN 8 AFOS BLOCK PRODUCT
      IER=0;ERROR RETURN
      CALL KSRCF(ID,JBUF,IER1)
      IF(IER1.NE.1) RETURN
      CALL RDBKF(0,JBUF,IER1)
      IF(IER1.NE.1) RETURN
      CALL UNPACK(JBUF,256,KBUF)
      DO 1 I=37,50
      IF(KBUF(I).EQ.12K) GO TO 2;LOOK FOR BEGINNING OF MESSAGE
 1    CONTINUE
      RETURN;NO 12K FOUND-ERROR
 2    CONTINUE
      ISTART=I+1;FIRST CHARACTER OF MESSAGE
      IF(KBUF(ISTART).EQ.40K) ISTART=ISTART+1
      NBYTES=0
      DO 3 I=ISTART,256
      IF(KBUF(I).EQ.203K) GO TO 55;END OF MESSAGE
      NBYTES=NBYTES+1
 3    IHOLD(NBYTES)=KBUF(I)
C READ IN ADDITIONAL BLOCKS
      IBLK=1;BLOCK COUNTER
 10   IBLK=IBLK+1;CURRENT BLOCK TO BE READ IN
      IF(IBLK.GT.8) RETURN;ERROR-END OF MESSAGE NOT FOUND AFTER 8 BLOCKS
      CALL NXBKF(JBUF,IER1)
      IF(IER1.NE.1) RETURN;ERROR-NEXT BLOCK NOT THERE
      CALL UNPACK(JBUF,256,KBUF)
      DO 4 J=5,256
      IF(KBUF(J).EQ.203K) GO TO 55;END OF MESSAGE
      NBYTES=NBYTES+1
 4    IHOLD(NBYTES)=KBUF(J)
      GOTO 10
 55   CONTINUE
C CHECK FOR CR/LF
      IF(IHOLD(NBYTES).EQ.0) NBYTES=NBYTES-1
      IF(IHOLD(NBYTES).EQ.12K) GO TO 56
      NBYTES=NBYTES+1
      IHOLD(NBYTES)=15K
      NBYTES=NBYTES+1
      IHOLD(NBYTES)=12K
 56   CALL PACK(IHOLD,NBYTES,IBUF)
      IER=1;SUCCESSFUL RETURN
      RETURN
      END
```

I. IDENTIFICATION

Module name: GCOORD
Date: October 10 1983
Function: Station information retrieval from STDIR.MS
Language: Fortran

II. FUNCTIONAL SUMMARY

GCOORD retrieves station information from STDIR.MS

III. CALLING METHOD

CALL GCOORD(ISTA,ALAT,ALONG,H,IER)

ISTA-4 byte array containing ASCII station ID. If the ID is only 3 characters long (as most FAA ID's) the fourth byte must be a space character
ALAT-Returned latitude of the station in floating point format
ALONG-Returned longitude of the station in floating point format
H-Altitude in feet ASL of the station
IER- If non-zero, station was not found. If equal to 0, successful return

```
      SUBROUTINE GCOORD(ISTA,ALAT,ALONG,H,IER5)
C     NREC=NUMBER OF RECORDS
C     LREC=LENGTH OF EACH RECORD (BYTES)
C     ISTAR=BYTE OF FIRST RECORD (0=BEGINNING)
C     IFLDP=WORD POINTER TO FIELD IN RECORD
C     IFLD=LENGTH OF FIELD IN BYTES
C     ITEST=ARRAY CONTAINING TEST FIELD
C     IAD=RETURNED TWO WORD ARRAY CONTAINING ADDRESS ITEST RECORD
C       SHOULD BEGIN AT-
C       IC= 1,2,3 IN SECOND WORD INDICATING RECORD WAS FOUND AND
C       IS IN ARRAY IC1,IC2, OR IC3
C     THOSE THREE ARRAYS SHOULD BE DIMENSIONED LREC/2 WORDS
      INTEGER JBUF(15)
      DIMENSION ITEST(20),IC1(20),IC2(20),IC3(20),IAD(2)
      INTEGER ISTA(2)
      DIMENSION IAD1(2),IAD2(2),IAD3(2)
      DIMENSION D1(2),D2(2)
      INTEGER D1,D2
      IER5=1
      ITEST(1)=ISTA(1)
      ITEST(2)=ISTA(2)
      DO 1 I=3,6
    1 ITEST(I)=20040K
      CALL GCHN(ICHN,IER)
      CALL OPENN(ICHN,"STDIR.MS",0,IER)
      NB=30
      CALL RDS(ICHN,JBUF,NB,IER)
      NREC=JBUF(1)
      LREC=JBUF(2)
      ISTAR=JBUF(3)
      IFLDP=JBUF(4)
      IFLD=(JBUF(5)-JBUF(4))*2
      IC=0
      IAD1(1)=0
      IAD1(2)=ISTAR
      CALL SPOS(ICHN,IAD1,IER)
      CALL ERROR(IER,'I1')
      CALL RDS(ICHN,IC1,LREC,IER)
      CALL ERROR(IER,'RDS - IC1')
      D2(1)=0
      D2(2)=LREC
      CALL DSUB(D2,D2,IAD1)
      CALL DMPY(D1,NREC,LREC)
      CALL DSUB(IAD2,D1,D2)
      CALL SPOS(ICHN,IAD2,IER)
      CALL ERROR(IER,'I2')
      CALL RDS(ICHN,IC2,LREC,IER)
      CALL ERROR(IER,'RDS-IC2')
      CALL BCOMP(IC1(IFLDP),ITEST,IFLD,IER1)
      IF(IER1.GT.1)GO TO 100
      CALL BCOMP(IC2(IFLDP),ITEST,IFLD,IER2)
      IF(IER2.NE.2)GO TO 125
    5 CALL DSUB(D1,IAD2,IAD1)
      CALL DDVD(INC,IR,D1,LREC)
      IF(INC.GE.32767)GO TO 900
      IF(INC.LT.1)GO TO 150
      INC=(INC-1)/2+1
      CALL DMPY(D1,INC,LREC)
      CALL DADD(IAD3,IAD1,D1)
      CALL SPOS(ICHN,IAD3,IER)
```

```
      CALL ERROR(IER,'I5')
      CALL RDS(ICHN,IC3,LREC,IER)
      CALL ERROR(IER,'I6')
      CALL BCOMP(IC3(IFLDP),ITEST,IFLD,IER3)
      IF(IER3.EQ.1)GO TO 50
      IF(IER3.EQ.2)GO TO 60
      IF(IER3.NE.3)GO TO 900
      IAD(1)=IAD3(1)
      IAD(2)=IAD3(2)
      IC=3
      GO TO 10
   50 IAD1(1)=IAD3(1)
      IAD1(2)=IAD3(2)
      GO TO 5
   60 IAD2(1)=IAD3(1)
      IAD2(2)=IAD3(2)
      IF(INC.EQ.1)GO TO 150
      GO TO 5
  100 IAD(1)=IAD1(1)
      IAD(2)=IAD1(2)
      IF(IER1.NE.3)GO TO 101
      IC=1
      IAD(1)=IAD1(1)
      IAD(2)=IAD1(2)
  101 GO TO 10
  125 D1(1)=0
      D1(2)=LREC
      CALL DADD(IAD,D1,IAD2)
      IF(IER2.NE.3)GO TO 126
      IAD(1)=IAD2(1)
      IAD(2)=IAD2(2)
      IC=2
  126 GO TO 10
  150 IAD(1)=IAD3(1)
      IAD(2)=IAD3(2)
      GO TO 10
  900 CALL ERROR(IER3,'IER3')
      IER=2
      CALL ERROR(IER,'TOO MANY RECORDS IN FILE')
      CALL EXIT
   10 CONTINUE
      IF(IC.EQ.0) GO TO 2
      CALL KLOSE(ICHN,IER)
      GOTO(31,32,33),IC
   31 IX=IC1(7)
      IY=IC1(6)
      IH=IC1(5)
      GO TO 35
   32 IX=IC2(7)
      IY=IC2(6)
      IH=IC2(5)
      GO TO 35
   33 IX=IC3(7)
      IY=IC3(6)
      IH=IC3(5)
   35 ALAT=IY/100.0
      ALONG=IX/100.0
      H=IH*3.2802399
      IER5=0
    2 CONTINUE
```

```
RETURN
END
```

I. IDENTIFICATION

Module name: INTERP
Date: October 10 1983
Function: Interpolate between two pressure levels in the vertical for a temperature
Language: Fortran

II. FUNCTIONAL SUMMARY

INTERP will interpolate between two reported pressure levels in a rawinsonde report to give the temperature at a given pressure level.

III. CALLING METHOD

CALL INTERP(P,T,Pi,Ti,NUM)

P-Floating point array of pressure levels from rawinsonde data (input)

T-Floating point array of temperatures from rawinsonde data (input)

Pi-Floating point pressure level at which a temperature is desired (input)

Ti-Floating point temperature at level Pi (output)

NUM-Input binary number of levels in arrays P and T

```
      SUBROUTINE INTERP(P,T,PSTOP,TSTOP,NUM)
C FIND T AT PSTOP
      DIMENSION P(1),T(1)
      DO 25 I=2,NUM
      P1=P(I-1)
      P2=P(I)
      P1=P1**0.287
      P2=P2**0.287
      T1=T(I-1)
      T2=T(I)
      A=(T2-T1)/(P2-P1)
      B=T1-A*P1
      START=P(I-1)
      DO 26 J=1,500
      IF(START.LT.PSTOP) GO TO 25
      TSTOP=A*(START**0.287)+B
 26   START=START-1
 25   CONTINUE
      RETURN
      END
```

I. IDENTIFICATION

Module name: IRTC
Date: October 10 1983
Function: Convert Radians to compass degrees
Language: Fortran

II. FUNCTIONAL SUMMARY

IRTC is a FORTRAN function that converts Radians to compass degrees

III. CALLING METHOD

IRTC is a FORTRAN Function

A=IRTC(R)

R=Radians

```
      INTEGER FUNCTION IRTC(RAD)
C CHANGE TRIG RADIANS TO COMPASS DEGREES
      IDEG=RAD*57.3+0.5
      IDEG=450-IDEG
      IF(IDEG.GT.360) IDEG=IDEG-360
      IRTC=IDEG
      RETURN
      END
```

I. IDENTIFICATION

Module name: CTR
Date: October 10 1983
Function: Convert compass degrees to radians
Language: Fortran

II. FUNCTIONAL SUMMARY

CTR is a FORTRAN function that converts compass degrees to radians

III. CALLING METHOD

CTR is a FORTRAN function

```
A=CTR(IR)
```

IR=Integer Compass Degrees

```
      REAL FUNCTION CTR(IDIR)
C CHANGE COMPASS DEGREES TO TRIG RADIANS
      IDEG=450-IDIR
      IF(IDEG.GE.360) IDEG=IDEG-360
      CTR=IDEG/57.3
      RETURN
      END
```

I. IDENTIFICATION

Module name:	CURV
Date:	October 10 1983
Function:	Fit a second order polynomial in Z variables to a set of data
Language:	Fortran

II. FUNCTIONAL SUMMARY

CURV uses the least squares technique to fit a set of data to a second order polynomial in Z variables. This routine is especially useful for non-linear interpolation of observed meteorological data.

Any number of X,Y,Z data points are accepted by this routine, but the mathematics require that at least 9 data points be input to the routine to get the high order terms of the polynomial. If less than 9 data points are entered, the routine will return a value of zero for the coefficients of the high order terms.

III. CALLING METHOD

CALL CURV(X,Y,Z,C,NUM)

X-Input array of floating point X coordinates. Any size dimension of the array is accepted by the routine.

Y-Input array of floating point Y coordinates. Any size dimension of the array is accepted by the routine.

Z-Input array of floating point observed values at coordinates X,Y. Z(N) is the observed value of some function at coordinates X(n),Y(n). Any size dimension of the array is accepted by the routine.

C-Output floating point array (dimension 25) with coefficients of the polynomial, as obtained in CURV

```
      SUBROUTINE CURV(X,Y,P,C,NUM)
C FIT P TO 2ED ORDER POLYNOMIAL
      DIMENSION X(1),Y(1),P(1),X1(200,25),Y1(200),C(25)
C
C
C
C
C
      DO 1 I=1,200
      DO 2 J=1,25
    2 X1(I,J)=0.0
    1 Y1(I)=0.0
      DO 3 I=1,NUM
      X1(I,1)=X(I)
      X1(I,2)=Y(I)
      X1(I,3)=X(I)*Y(I)
      X1(I,4)=X(I)*X(I)
      X1(I,5)=Y(I)*Y(I)
      X1(I,6)=X(I)*X(I)*Y(I)
      X1(I,7)=X(I)*Y(I)*Y(I)
      X1(I,8)=X(I)*X(I)*Y(I)*Y(I)
    3 Y1(I)=P(I)
      CALL LSTSQ(X1,Y1,C,NUM,8)
      RETURN
      END
```

I. IDENTIFICATION

Module name: SCURV
Date: October 10 1983
Function: Evaluate the polynomial fitted by CURV at coordinates X,Y
Language: Fortran

II. FUNCTIONAL SUMMARY

SCURV will evaluate the second order polynomial in two variables which was fit by subroutine CURV at any given coordinates X,Y. See CURV for additional details

III. CALLING METHOD

CALL SCURV(X,Y,Z,C)

X-Input floating point X coordinate to evaluate polynomial at

Y-Input floating point Y coordinate to evaluate polynomial at

Z-Output floating point value of the polynomial at coordinates X,Y

C-Input floating point array (dimension 25) with coefficients of the polynomial, as obtained in CURV

```
      OVERLAY INDX3
      SUBROUTINE SCURV(X,Y,Z,C)
      DIMENSION C(1)
C SOLVE CURVE FITTED BY CURV.FR
      Z=C(1)
      Z=Z+C(2)*X
      Z=Z+C(3)*Y
      Z=Z+C(4)*X*Y
      Z=Z+C(5)*X*X
      Z=Z+C(6)*Y*Y
      Z=Z+C(7)*X*X*Y
      Z=Z+C(8)*X*Y*Y
      Z=Z+C(9)*X*X*Y*Y
      RETURN
      END
```

I. IDENTIFICATION

Module name: LSTSQ
Date: October 10 1983
Function: Setup and solve a least-squares matrix
Language: Fortran

II. FUNCTIONAL SUMMARY

LSTSQ is a routine that allows the user to calculate a least squares fit to a set of data, with up to 24 independent variables. A solution is found with the form

Y=C(1) + C(2)*X(1) + C(3)*X(2) + C(4)*X(3) + ... + C(N+1)*X(N)

III. CALLING METHOD

CALL LSTSQ(X,Y,C,NUM,N)

X-Independent values. Must be a floating point array dimensioned (200,25). Up to 200 observations can be entered, each with up to 24 elements. (Input)

Y-Dependent values. Must be a floating point array dimensioned to 200. This contains the observed values for the elements that you want to derive a relation for. (Input)

C-Coefficient Matrix. Must be a floating point array dimensioned to 25. This contains the calculated coefficients for the least squares equation. There will be N+1 coefficients in the returned array. (Output)

NUM-Integer number of observations in X and Y arrays, up to 200 (Input)

N-Integer number of elements in each observation, up to 24 (Input)

```
      SUBROUTINE LSTSQ(X,Y,C,NUM,IORD)
      DIMENSION A(25,26),C(25),WORK(25),X(200,25),Y(200)
      DO 1 I=1,25
      C(I)=0
      WORK(I)=0
      DO 1 J=1,26
 1    A(I,J)=0
C SET UP LSTSQ MATRIX
      IORD1=IORD+1
      IORD2=IORD+2
      IF(IORD1.GT.25) RETURN
      DO 3 I=1,NUM
      WORK(1)=1.0
      DO 2 J=2,IORD1
      J1=J-1
 2    WORK(J)=X(I,J1)
      DO 5 K=1,IORD1
      DO 4 J=1,IORD1
  4   A(K,J)=A(K,J)+WORK(K)*WORK(J)
C SET UP AUGMENTED MATRIX
 5    A(K,IORD2)=A(K,IORD2)+WORK(K)*Y(I)
 3    CONTINUE
C SOLVE
      DO 6 I=1,IORD1
       ADIV=A(I,I)
      DO 7 J=1,IORD2
 7    A(I,J)=A(I,J)/ADIV
      DO 8 K=1,IORD1
      IF(K.EQ.I) GO TO 8
      ADIV=A(K,I)
      DO 9 J=1,IORD2
 9    A(K,J)=A(K,J)-ADIV*A(I,J)
 8    CONTINUE
 6    CONTINUE
      DO 10 J=1,IORD1
 10   C(J)=A(J,IORD2)
      RETURN
      END
```

I. IDENTIFICATION

Module name: GDEC
Date: October 10 1983
Function: Convert unpacked ASCII to floating point
Language: Fortran

II. FUNCTIONAL SUMMARY

GDEC converts unpacked ASCII characters to floating point representation

III. CALLING METHOD

CALL GDEC(IBUF,AVAL,IER)

IBUF-Input array containing unpacked ASCII representation of floating point value. The string must be terminated by a null byte or a space character.

AVAL-Output floating point value.

IER-Error return. If zero, indicates successful conversion. A non-zero return indicates that non-numeric characters were found in the unpacked ASCII array

GDEC will attempt to convert numbers greater than 32000.0 or smaller than -32000.0, resulting in a runtime error 5.

```
      SUBROUTINE GDEC(IHOLD,AVAL,IER)
      INTEGER IHOLD(1),IHOLD1(5)
      IER=1
C SEARCH FOR END OF STRING
      DO 10 I=1,15
      IF(IHOLD(I).EQ.0.OR.IHOLD(I).EQ.32) GO TO 11
 10   CONTINUE
      RETURN
 11   NCHAR=I-1
C SEARCH FOR MINUS ,PLUS AND DECIMAL
      INEG=1
      IDECPOS=0
      DO 1 I=1,NCHAR
      IF(IHOLD(I).EQ.46) IDECPOS=I
      IF(IHOLD(I).EQ.46) IHOLD(I)=48
      IF(IHOLD(I).EQ.45) INEG=(-1)
      IF(IHOLD(I).EQ.45) IHOLD(I)=48
 1    IF(IHOLD(I).EQ.43) IHOLD(I)=48
C GET INTEGER PORTION
      I1=IDECPOS-1
      DO 8 I=1,5
 8    IHOLD1(I)=32
      IF(IDECPOS.EQ.0) I1=5
      DO 2 I=1,I1
 2    IHOLD1(I)=IHOLD(I)
      CALL DECODE(IHOLD1,IVAL,IER)
      IF(IER.NE.0) RETURN
      IF(IDECPOS.EQ.0) AVAL=IVAL*INEG
      IF(IDECPOS.EQ.0) RETURN
C GET DECIMAL PORTION
      EX=1.0
      I1=IDECPOS+1
      AVAL=IVAL
      DO 3 I=I1,10
      IF(IHOLD(I).EQ.32) GO TO 5;RETURN
      INUM=IHOLD(I)-48
      AVAL=AVAL+INUM/(10.0**EX)
      EX=EX+1
 3    CONTINUE
      IER=1
      RETURN
 5    AVAL=AVAL*INEG
      RETURN
      END
```

I. IDENTIFICATION

Module name: ENCODE
Date: October 10 1983
Function: Convert binary to ASCII representation
Language: Fortran

II. FUNCTIONAL SUMMARY

ENCODE takes an input binary value and returns it in packed ASCII representation

III. CALLING METHOD

CALL ENCODE(IBUF,IVAL,NB)

IBUF-Output array to recieve packed ASCII representation of binary value. The string is terminated by a null byte.

IVAL-Input binary value. Allowable values -32000 to 32000

NB-Output number of bytes (characters) in the ASCII representation

```
      SUBROUTINE ENCODE(ID,INUM,NB)
      INTEGER ID(1),IHOLD(6)
      INUMSV=INUM
      ICD=0
      DO 3 I=1,6
 3    IHOLD(I)=0
C CONVERT INTEGER FORM /INUM/ TO ASCII FORM WITH LAST WORD BINARY 0
      ICOUNT=0
      IF(INUM.GE.0) GO TO 4
      ICOUNT=1
      IHOLD(1)=45
      INUM=INUM*(-1)
C CHECK FOR EQ TO ZERO
 4    CONTINUE
      IF(INUM.NE.0) GO TO 5
      ID(1)=ISHFT(48,8)
      ID(2)=0
      NB=1
      RETURN
 5    CONTINUE
      DO 1 I=1,5
      I1=10**(5-I)
      ITEST=INUM-I1
      IF(ITEST.LT.0.AND.ICD.EQ.0) GO TO 1
      ICD=1
      ICOUNT=ICOUNT+1
      IHOLD(ICOUNT)=INUM/I1+48
      INUM=INUM-I1*(IHOLD(ICOUNT)-48)
 1    CONTINUE
      CALL PACK(IHOLD,6,ID)
      ID(4)=0
      NB=ICOUNT
      INUM=INUMSV
      RETURN
      END
```

I. IDENTIFICATION

Module name: DECODE
Date: October 10 1983
Function: Convert unpacked ASCII to binary
Language: Fortran

II. FUNCTIONAL SUMMARY

DECODE converts unpacked ASCII characters to binary integer representation

III. CALLING METHOD

CALL DECODE(IBUF,IVAL,IER)

IBUF-Input array containing unpacked ASCII representation of binary value. The string must be terminated by a null byte or a space character.

IVAL-Output binary integer value.

IER-Error return. If zero, indicates successful conversion. A non-zero return indicates that non-numeric characters were found in the unpacked ASCII array

DECODE will attempt to convert numbers greater than 32000 or smaller than -32000, resulting in a runtime error 5.

```
      SUBROUTINE DECODE(IHOLD,IVAL,IER)
      INTEGER IHOLD(1)
      IER=1
C SEARCH FOR END OF STRING
      DO 4 I=1,10
      IF(IHOLD(I).EQ.0.OR.IHOLD(I).EQ.32) GO TO 5
 4    CONTINUE
      RETURN
 5    NCHAR=I-1
C SEARCH FOR MINUS SIGN
      INEG=1
      DO 1 I=1,NCHAR
      IF(IHOLD(I).EQ.43) IHOLD(I)=32
      IF(IHOLD(I).EQ.45) INEG=(-1)
 1    IF(IHOLD(I).EQ.45) IHOLD(I)=32
C CHECK FOR VALID DIGITS
      DO 2 I=1,NCHAR
      IF(IHOLD(I).EQ.32) GO TO 2
      IF(IHOLD(I).LT.48.OR.IHOLD(I).GT.57) RETURN
 2    CONTINUE
C ALL DIGITS OK...COMPUTE VALUE
      IEX=0
      IVAL=0
      DO 3 I=1,NCHAR
      I1=NCHAR-I+1
      IF(IHOLD(I1).EQ.32) GO TO 3
      IVAL1=(IHOLD(I1)-48)
      IVAL=IVAL+IVAL1*10**IEX
      IEX=IEX+1
 3    CONTINUE
      IVAL=IVAL*INEG
      IER=0
      RETURN
      END
```

I. IDENTIFICATION

Module name: FCODE
Date: October 10 1983
Function: Convert floating point to ASCII representation
Language: Fortran

II. FUNCTIONAL SUMMARY

FCODE takes an input floating point value and returns it in packed ASCII representation according to a specified format

III. CALLING METHOD

CALL FCODE(A,IBUF,NDEC,NB)

A-Input floating point number to be represented in ASCII

IBUF-Output array to revieve the packed ASCII characters. The character string is terminated by a null byte

NDEC-Input number of decimal places to carry in the ASCII representation. The maximum allowed is 6.

NB-The number of bytes (characters) output in the IBUF array

FCODE rounds the input floating point number to the number of decimal places specified. FCODE can only handle numbers from -32000.0 to 32000.0.

```
      SUBROUTINE FCODE(AVAL,IBUF,NDEC,NB)
C   ENCODE A DECIMAL AS AN ASCII CHARACTER
C LIMIT OF +/- 32000 FOR VALUE OF AVAL
C NDEC=NUMBER OF DECIMAL PLACES TO CARRY
      INTEGER IBUF(1),IHOLD(10),IHOLD1(10),IBUF1(20),IBUF2(20)
      NCHAR=0
      IINT=AVAL
      DEC=AVAL-IINT
      CALL ENCODE(IHOLD,IINT,NCHAR)
C ENCODE DECIMAL PORTION...ROUND TO NEAREST SIGNIFICANT DIGIT
      I1=DEC*(10**NDEC)+0.5
      IF(I1.LT.0) I1=I1*(-1)
      CALL ENCODE(IHOLD1,I1,NCHAR1)
      CALL UNPACK(IHOLD1,20,IBUF1)
      DO 6 I=1,15
      IF(NCHAR1.GE.NDEC) GO TO 7
C FILL IN A PRECEDING ZERO DIGIT
      DO 8 J=1,19
      J1=21-J
      J2=20-J
  8   IBUF1(J1)=IBUF1(J2)
      IBUF1(1)=48
  6   NCHAR1=NCHAR1+1
  7   CONTINUE
C ADD PRECEDING DECIMAL POINT
      DO 9 J=1,19
      J1=21-J
      J2=20-J
 9    IBUF1(J1)=IBUF1(J2)
      IBUF1(1)=46
      NCHAR1=NCHAR1+1
C UNPACK INTEGER PORTION
      CALL UNPACK(IHOLD,20,IBUF2)
C     REPACK
      DO 10 I=1,NCHAR1
      I1=NCHAR+I
 10   IBUF2(I1)=IBUF1(I)
      ISTART=I1+1
      DO 14 I=ISTART,20
 14   IBUF2(I)=0
      CALL PACK(IBUF2,20,IBUF)
      NB=NCHAR(NCHAR1
      RETURN
      END
```

I. IDENTIFICATION

Module name: PRVDAY
Date: October 10 1983
Function: Calculate the date of the previous day
Language: Fortran

II. FUNCTIONAL SUMMARY

PRVDAY calculates the date of the previous day, given the current date

III. CALLING METHOD

CALL PRVDAY(IDATE,JDATE)

IDATE-Input three word array containing date. Word 1 is the binary month, word 2 the date, and word 3 the year

JDATE-Output three word array containing previous days date. Word 1 is the binary month, word 2 the date, and word 3 the year.

```
C SUBROUTINE PRVDAY...FIND PREVIOUS DATE
      SUBROUTINE PRVDAY(IDATE,JDATE)
      INTEGER IDATE(3),JDATE(3)
      COMMON/PRVD/IMO(12)
      DATA IMO/31,28,31,30,31,30,31,31,30,31,30,31/
      ITEST=MOD(IDATE(3),4)
      IF(ITEST.EQ.0) IMO(2)=29
      DO 1 I=1,3
 1    JDATE(I)=IDATE(I)
      JDATE(2)=JDATE(2)-1
      IF(JDATE(2).GE.1) RETURN;STILL IN SAME MONTH
C FIND PREVIOUS MONTH
      JDATE(1)=JDATE(1)-1
      IF(JDATE(1).LT.1) JDATE(3)=JDATE(3)-1
      IF(JDATE(1).LT.1) JDATE(1)=12
      I1=JDATE(1)
      JDATE(2)=IMO(I1)
      RETURN
      END
```

I. IDENTIFICATION

Module name: NXTDAY
Date: October 10 1983
Function: Calculate the date of the next day
Language: Fortran

II. FUNCTIONAL SUMMARY

NXTDAY calculates the date of the next day, given the current date

III. CALLING METHOD

CALL NXTDAY(IDATE,JDATE)

IDATE-Input three word array containing date. Word 1 is the binary month, word 2 the date, and word 3 the year

JDATE-Output three word array containing next days date. Word 1 is the binary month, word 2 the date, and word 3 the year.

```
C SUBROUTINE NXTDAY...GIVEN CURRENT DATE...FIND DATE OF NEXT DAY
      SUBROUTINE NXTDAY(IDATE,JDATE)
C IDATE...3 WORD ARRAY CONTAIN MO/DATE/YR
C JDATE...3 WORD ARRAY CONTAINING NEXT DAYS MO/DATE/YR
      INTEGER IDATE(3),JDATE(3)
      COMMON/NDAS/IMO(12)
      DATA IMO/31,28,31,30,31,30,31,31,30,31,30,31/
      ITEST=MOD(IDATE(3),4);CHECK FOR LEAP YEAR
      IF(ITEST.EQ.0) IMO(2)=29
      DO 1 I=1,3
 1    JDATE(I)=IDATE(I)
      JDATE(2)=JDATE(2)+1;INCREMENT DAY
      I1=IDATE(1)
      IF(JDATE(2).LE.IMO(I1)) RETURN;STILL IN SAME MONTH
      JDATE(2)=1
      JDATE(1)=JDATE(1)+1;INCREMENT MONTH
      IF(JDATE(1).LT.13) RETURN;STILL IN SAME YEAR
      JDATE(1)=1
      JDATE(3)=JDATE(3)+1;INCREMENT YEAR
      RETURN
      END
```

I. IDENTIFICATION

Module name: AMPM
Date: October 10 1983
Function: Convert GMT time to Local time
Language: Fortran

II. FUNCTIONAL SUMMARY

AMPM returns both the GMT system time and the computer local time.

III. CALLING METHOD

CALL AMPM(IDATE,ITIME,JDATE,JTIME,IZN)

IDATE-Three word array containing output GMT date. Word 1 contains binary month, word 2 date, word 3 binary year

ITIME-Three word array containing output GMT time. Word 1 contains binary hours, word 2 minutes, word 3 seconds

JDATE-Three word array containing output local date. Word 1 contains binary month, word 2 date, word 3 binary year

JTIME-Three word array containing output local time. Word 1 contains binary hour, word 2 minutes, word 3 seconds

IZN=Input ASCII time zone. (PS,PD,MS,MD,CS,CD,ES,ED)

```
      OVERLAY STP2
      SUBROUTINE AMPM(ZDATE,ZTIME,LDATE,LTIME,TMZN)
C SUBROUTINE TO GET SYSTEM DATE...AND CONVERT TO LOCAL TIME
C ALSO PREPARES A LINE CONTAINING ASCII DATE/TIME
C INPUT...TMZN...=MD MS CD CS ED ES PS PD
C OUTPUT...ZDATE...Z DATE   LDATE...LOCAL DATE
C          ZTIME...Z TIME   LTIME...LOCAL TIME
      INTEGER ZDATE(3),ZTIME(3),LDATE(3),LTIME(3),TMZN,OFFSET
      INTEGER KDATE(3)
      IT=IAND(TMZN,377K);IS IT DAYLIGHT OR STANDARD TIME?
      ITFLAG=0;STANDARD TIME
      IF(IT.EQ.68) ITFLAG=1;DAYLIGHT TIME
C FIND ZONE
      IT1=ISHFT(IAND(TMZN,177400K),-8)
      IOFF=-7
      IF(IT1.EQ.69) IOFF=-5
      IF(IT1.EQ.67) IOFF=-6
      IF(IT1.EQ.80) IOFF=-8
      IOFF=IOFF+ITFLAG;TIME CORRECTED FOR STANDARD/DAYLIGHT TIME
C GET SYSTEM DATE/TIME
      CALL DATE(ZDATE,IER)
      CALL TIME(ZTIME,IER)
C ADD OFFSET TO HOUR. IF RESULTS ARE LESS THAN 0...MUST CHANGE
C DATE
      DO 1 I=1,3
      LDATE(I)=ZDATE(I)
 1    LTIME(I)=ZTIME(I)
      LTIME(1)=LTIME(1)+IOFF
      IF(LTIME(1).GE.0) RETURN;DONE
C MUST CHANGE DATE
      LTIME(1)=LTIME(1)+24
      CALL PRVDAY(LDATE,KDATE)
      DO 2 I=1,3
 2    LDATE(I)=KDATE(I)
      RETURN
      END
```

NOAA SCIENTIFIC AND TECHNICAL PUBLICATIONS

The National Oceanic and Atmospheric Administration was established as part of the Department of Commerce on October 3, 1970. The mission responsibilities of NOAA are to assess the socioeconomic impact of natural and technological changes in the environment and to monitor and predict the state of the solid Earth, the oceans and their living resources, the atmosphere, and the space environment of the Earth.

The major components of NOAA regularly produce various types of scientific and technical information in the following kinds of publications:

PROFESSIONAL PAPERS — Important definitive research results, major techniques, and special investigations.

CONTRACT AND GRANT REPORTS — Reports prepared by contractors or grantees under NOAA sponsorship.

ATLAS — Presentation of analyzed data generally in the form of maps showing distribution of rainfall, chemical and physical conditions of oceans and atmosphere, distribution of fishes and marine mammals, ionospheric conditions, etc.

TECHNICAL SERVICE PUBLICATIONS — Reports containing data, observations, instructions, etc. A partial listing includes data serials; prediction and outlook periodicals; technical manuals, training papers, planning reports, and information serials; and miscellaneous technical publications.

TECHNICAL REPORTS — Journal quality with extensive details, mathematical developments, or data listings.

TECHNICAL MEMORANDUMS — Reports of preliminary, partial, or negative research or technology results, interim instructions, and the like.

Information on availability of NOAA publications can be obtained from:

ENVIRONMENTAL SCIENCE INFORMATION CENTER
ENVIRONMENTAL DATA AND INFORMATION SERVICE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
U.S. DEPARTMENT OF COMMERCE

Rockville, MD 20852