SM23/RR 3/5

stern Region Computer Programs
blems NWS ERCP - No. 8

DECODERS FOR FRH, FTJ AND FD PRODUCTS

Scientific Services Division
Eastern Region Headquarters
February 1983

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

NOAA Technical Memorandum
National Weather Service, Eastern Region Computer Programs and Problems

The Eastern Region Computer Programs and Problems (ERCP) series is a subset of the Eastern Region Technical Memorandum series. It will serve as the vehicle for the transfer of information about fully documented AFOS application programs. The format of ERCP - No. 1 will serve as the model for future issuances in this series.

1 An AFOS version of the Flash Flood Checklist. Cynthia M. Scott, March 1981. (PB81 211252).

2 An AFOS Applications Program to Compute Three-Hourly Stream Stages. Alan P. Blackburn, September 1981. (PB82 156886).

3 PUPPY (AFOS iel P. Provost, December 198

4 Special Sear April 1982.

5 Conversion c October 1982.

6 Real-Time Qu January 1983.

7 Automated Ho put. Lawrence Cedrone, Jan

QC
874.3
.U63
no.8

NOAA Eastern Region Computer Programs and Problems - No. 8

DECODERS FOR FRH, FTJ AND FD PRODUCTS

Cynthia M. Scott

Scientific Services Division
Eastern Region Headquarters

February 1983

UNITED STATES DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary
/ **National Oceanic and Atmospheric Administration**
John V. Byrne, Administrator
/ National Weather Service
Richard E. Hallgren,
Acting Assistant Administrator

DECODERS FOR FRH, FTJ AND FD PRODUCTS

Cynthia M. Scott
Scientific Services Division
National Weather Service Eastern Region
Garden City, New York

I. General Information

A. Summary

Numerical output from the LFM and trajectory models is available in AFOS in the FRH, FTJ and various FD products. This information has many potential uses in local meteorological applications programs. The three basic subroutines in this package will extract any desired value from these products. Additional subroutines are available to minimize database access when data for many stations is being extracted from the bulletin-format products (FRH and FTJ).

The Appendix describes a useful program for working with database products that have been unpacked into arrays. It prints out the arrays in a pseudo-FPRINT form on the Dasher.

B. Environment

All the routines were written in Data General's FORTRAN IV and will run on the ECLIPSE. AFOS must be up for the database access to work.

C. References

Reap, R. M., 1978: The Trajectory (TRAJ) Model. <u>NWS Technical Procedures Bulletin</u> No. 225, National Oceanic and Atmospheric Administration, Department of Commerce, 13 pp.

Sadowski, A. and R. Hollern, 1981: FOUS 60-78 Bulletins. <u>NWS Technical Procedures Bulletin</u> No. 294, National Oceanic and Atmospheric Administration, Department of Commerce, 11 pp.

II. Application

A. Complete Program Description

The three basic subroutines are called GETPARFRH, GETPARFTJ and GETPARFD. Although they differ in detail, their basic principle of operation is the same. All of them work on AFOS database products (FRH, FTJ and FDx) that have been placed in unpacked arrays. These products are all formatted, so there is a unique data location corresponding to a given time, parameter type (e.g., thickness or temperature), and except for FRH, level in the atmosphere. The subroutines take the specified parameter type, time and level information, convert it to row and column numbers, go to the specified location and pick the proper value out of the upacked array. The field size (or number of array words to pick out) is set by the parameter type. (Of course, values for some parameters are not available for every time period, such as the vertical velocity on the FRH or the temperature in the FTJ. Error messages alert the programmer if non-existent information has been requested. (A slight exception to this is the K index in the FTJ product. GETPARFTJ will retrieve it no matter which of the three levels is specified.)

At this point, the data extracted from the upacked AFOS product consists of a small array of ASCII characters, not a numerical value that could be used in further calculations. The subroutine NUMVALUE takes this array and converts it into a real number. Finally, implied digits are restored to the data. (The returned real value can be IFIX'd if integers are required.)

GETPARFRH, GETPARFTJ and GETPARFD all return one piece of data per call. When assembling data for many stations, the subroutines could be used over and over in a loop setup. This is fine for the individual FD products, but could pose a problem with the bulletin-formatted FRH and FTJ. It isn't very efficient to pull a large product out of the database, extract a value for one station, and then possibly do it all over again with the same product for the next station. The stations should be sorted so that each bulletin is pulled out only once, and data for all the relevant stations is extracted at that time. Two "envelope" subroutines called EXTRFRH and EXTRFTJ were written to do this sorting--all the calling program need do is supply an array of station id's. EXTRFRH and EXTRFTJ use GETPARFRH and GETPARFTJ to extract the data and return arrays of values for the specified time, parameter type, and level for the input stations.

Each of the subroutines has a logical error return set to TRUE in cases where non-existent information was requested or if NUMVALUE has found characters with no numerical meaning (usually caused by a garbled product). The calling program must be written to handle the error return and missing data properly--none of the subroutines will stop on their own.

B. Machine Requirements

Since all the programs described in this CP are subroutines (except for PRTPRODUCT in the Appendix), core and disk space requirements and run times can't really be determined. These factors depend on how large the main program is and what it is supposed to do. But some estimates can be made: using subroutine GETPARFD will add about 2000 words to a save file; GETPARFRH, about 3300 words; and GETPARFTJ, about 3800. Subroutines EXTRFRH and EXTRFTJ are much more complex and will add from 9000 to 9600 words. Some small testing programs for the basic subroutines (pulling out 1-3 pieces of data for 1 station) run in about 30 seconds. Programs testing EXTRFRH and EXTRFTJ (extracting 1 piece of data for 6 stations) take from 40 seconds to a minute. (These times include time to perform Dasher output--also AFOS was moderately busy at the time.)

No overlays or channels are used by the subroutines.

C. Structure of Software

GETPARFD, GETPARFRH and GETPARFTJ are quite straightforward. See Figure 1 for a flow chart for subroutine EXTRFRH. EXTRFTJ is similar but also has to accomodate bulletins of varying length and the fact that 850 mb data is not produced for some high altitude stations.

D. Database

FRH, FTJ and FD products are copied from the database by subroutine GETPRODUCT within subroutines EXTRFRH and EXTRFTJ. GETPRODUCT or a similar routine should be called seperately before using GETPARFD, GETPARFRH or GETPARFTJ. GETPRODUCT uses standard BG.LB routines.

III. Procedures

A. How to Use the Subroutines

Subroutines GETPARFD, GETPARFRH and GETPARFTJ all take AFOS products in unpacked array form as input. Therefore subroutine GETPRODUCT or an equivalent database extraction routine must be called first. The subroutine calls are:

```
CALL GETPRODUCT(KEY, HOLDER, EXTIME, ERROR)
```

where KEY is an integer input array holding an AFOS key
HOLDER is an integer output array for the unpacked AFOS product.
EXTIME is an integer input containing the difference

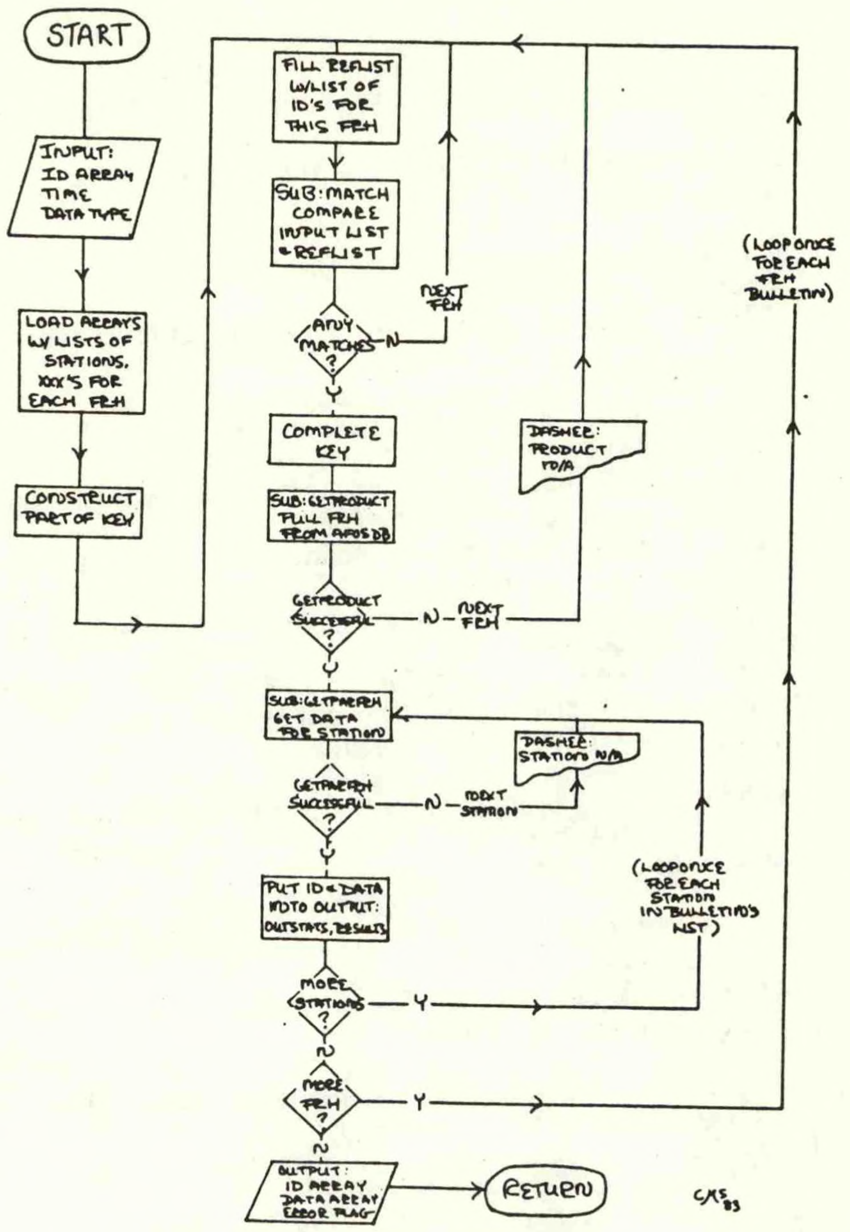
START
INPUT: ID ARRAY TIME DATA TYPE
LOAD ARRAYS W/ LISTS OF STATIONS, XXX'S FOR EACH FRH
CONSTRUCT PART OF KEY
FILL REFLIST W/LIST OF ID'S FOR THIS FRH
SUB: MATCH COMPARE INPUT LIST & REFLIST
ANY MATCHES ?
N
NEXT FRH
Y
COMPLETE KEY
SUB: GETPRODUCT PULL FRH FROM AFOS DB
GETPRODUCT SUCCESSFUL ?
N
NEXT FRH
DASHER: PRODUCT N/A
Y
SUB: GETPARFRH GET DATA FOR STATION
GETPARFRH SUCCESSFUL ?
N
NEXT STATION
DASHER: STATION N/A
Y
PUT ID & DATA INTO OUTPUT: OUTSTATS, RESULTS
MORE STATIONS ?
Y
(LOOP ONCE FOR EACH STATION IN BULLETIN'S LIST)
N
MORE FRH ?
Y
(LOOP ONCE FOR EACH FRH BULLETIN)
N
OUTPUT: ID ARRAY DATA ARRAY ERROR FLAG
RETURN
CMS 83

Figure 1. Flow chart for EXTRFRH.

in minutes between the current time and the product time beyond which the product will be considered old. (Use any negative number to omit the time check.)
ERROR is a logical error return.

and

```
CALL GETPARFD(UPBUFFER, LEVEL, PARAM, VALUE, ERROR)
CALL GETPARFRH(UPBUFFER, STATION, HOUR, PARAM, VALUE,
ERROR)
CALL GETPARFTJ(UPBUFFER, STATION, HOUR, LEVEL, PARAM,
VALUE, ERROR)
```

where UPBUFFER is an integer input array holding the unpacked AFOS product.
STATION is an integer input array holding the station id.
PARAM is an integer input indicating the type of data desired, e. g. "TT" for temperature. Parameter values are listed in the subroutine source files.
HOUR is an integer input indicating the valid time of the desired data, e. g. 36 for 36 hours.
LEVEL is an integer input indicating the level in the atmosphere of the desired data (in millibars for FTJ, thousands of feet for FD), e. g. 850 for 850 mb or 30 for 30,000 feet.
VALUE is a real (except for GETPARFD, where it is integer) output containing the extracted data.
ERROR is a logical error return.

EXTRFRH and EXTRFTJ call GETPRODUCT themselves, so a separate call is not needed. Their calling statements should look like:

```
CALL EXTRFRH(INSTATS, OUTSTATS, NSTATS, HOUR, PARAM,
RESULTS, ERROR)
CALL EXTRFTJ(INSTATS, OUTSTATS, NSTATS, HOUR, LEVEL,
PARAM, RESULTS, ERROR)
```

where INSTATS is a two-dimensional input array of station id's for which data is desired.
OUTSTATS is a two-dimensional output array of station id's for which data was found.
NSTATS is an integer input giving the number of stations in INSTATS.
HOUR (same as in previous subroutines)
LEVEL (same as in previous subroutines)
PARAM (same as in previous subroutines)

RESULTS is a real output array containing the extracted data. The order of the values corresponds to the order of the station id's in OUTSTATS.
ERROR is a logical error return.

B. Cautions and Restrictions

1. EXTRFRH will handle stations in FRH bulletins 60 through 73 only. To include other bulletins, their station id's and their key endings would have to be added to matrix IFRH6073 and array IXXLIST, respectively, and the number of times the main loop executes would have to be increased from 14.

2. The calling program must be able to properly handle any error returns and missing data, since none of the subroutines will stop on their own. Remember that the error returns are logical, so if an error occurs they are set to TRUE. If the returns are read as integers, TRUE (177777 octal) comes out as -1, so the typical successful return of 1 does not apply here.

3. If any format changes are made to the FRH, FTJ or FD products, the subroutines will need to be updated.

4. IMPORTANT...In GETPARFRH, the boundary layer temperature was used in assigning the implied digit to the thickness. This may result in bad thicknesses for stations in mountainous areas under certain conditions.

C. Complete Program Description

Complete listings of the subroutines follow the Appendix.

APPENDIX

I. General Information

A. Summary

This program extracts a desired product from the database and prints it out on the Dasher in a format similar to that of the FPRINT command.

B. Environment

The program is written in Data General's FORTRAN IV and will run on the ECLIPSE with AFOS up.

II. Application

A. Complete program description

Programming using extracted database products is sometimes difficult. The BG.LB routines will extract any desired product and unpack it into an array. But to use the data, its exact location has to be determined. SAVEing the product and FPRINTing it to find locations can give misleading results since the SAVE process works differently.

The program PRTPRODUCT was written to help this situation. It will extract any desired alphanumeric product from the database using subroutine GETPRODUCT and print it out word by word on the Dasher. The format is similar to FPRINT--rows and columns of words--except that the numbering is decimal and there are no ASCII equivalents printed (these may be added at a later date). Locations can then be easily determined by using a table of ASCII equivalents.

B. Machine requirements

PRTPRODUCT takes up 28 blocks of disk space and will run in 17K. Run time is dependent on AFOS traffic and the length of the desired product, but it generally takes under 30 seconds to begin printing out the product.

C. Database

Subroutine GETPRODUCT uses standard BG.LB routines to extract alphanumeric products from the database.

III. Procedures

A. Using the program

Running the program is very simple. Install PRTPRODUCT on DP0 or link it down to DP0F. At the Dasher, type PRTPRODUCT to start the program. The program will ask you to type in an AFOS key. The output product will be printed on the Dasher. See Figure A-1 for a sample run.

B. Cautions and Restrictions

PRTPRODUCT can handle products up to 15 blocks long. If the product requested is longer, the program will return an error message and stop. Products not in the database are handled similarly.

C. Complete Program Listing

A complete listing of PRTPRODUCT follows the listings of the subroutines covered in the main part of this CP.

```
PRTPRODUCT
TYPE THE KEY OF PRODUCT YOU WISH TO PRINT OUT
ERHOWFLGA
          0        1        2        3        4        5        6        7        8        9
  0     105 E      0      106 F    120 P    125 U    123 S     71 9     61 /     40      113 K
  1     127 W    102 B    103 C     40       61       71       60       66       60       60
  2      15       15       12       40       40       40       40       40       40      114 L
  3     107 G    101 A     40       15       15       12      116      105      127       40
  4     131      117      122      113       40      101      116      104       40      126
  5     111      103      111      116      111      124      131       40      106      117
  6     122      105      103      101      123      124       15       15       12      116
  7     101      124      111      117      116      101      114       40      127      105
  8     101      124      110      105      122       40      123      105      122      126
  9     111      103      105       40      116      105      127       40      131      117
 10     122      113       54       40      116      131       15       15       12       64
 11      61       65       40      101      115       40      105      123      124       40
 12     127      105      104      116      105      123      104      101      131       40
 13     112      101      116       40       61       71       40       61       71       70
 14      63       15       15       12      103 C    114 L    105 E    101 A    122 R     40
 15     101 A    116 N    104 D     40      102 B    111 /    124 T    124 T    105 E    122 R
 16      40      103 C    117 O    114 L    104 D     40      124 T    117 O    104 D    101 A
 17     131 Y     40      110      111      107      110       40      116      105      101
 18     122       40       62       60       56       40       40      123      124      122
 19     117      116      107       40      107      125      123      124      131       40
 20     116      117      122      124      110      127      105      123      124      105
 21     122      114      131       15       15       12      127      111      116      104
 22     123       40       62       65       40      124      117       40       63       65
 23      40      115      120      110       56       40       40      127      111      116
 24     104       40      103      110      111      114      114       40      106      101
 25     103      124      117      122       40       55       61       64       40      104
 26     105      107      122      105      105      123       56       40       40      124
 27     117      116      111      107      110      124       55       55      123      124
 28     122      117      116      107       15       15       12      107      125      123
 29     124      131       40      116      117      122      124      110      127      105
 30       NMCOWFLGA
 31       FPUS91 KWBC 190600
 32            LGA
 33       NEW YORK AND VICINITY FORECAST
 34       NATIONAL WEATHER SERVICE NEW YORK, NY
 35       415 AM EST WEDNESDAY JAN 19 1983
 36       CLEAR AND BITTER COLD TODAY HIGH NEAR 20.  STRONG GUSTY NORTHWESTERLY
 37       WINDS 25 TO 35 MPH.  WIND CHILL FACTOR -14 DEGREES.  TONIGHT--STRONG
 38       GUSTY NORTHWESTERLY WINDS 25 TO 35 MPH DIMINISHING TO 15 MPH BY
 39       MORNING.  CLEAR STEADY OR SLOWLY RISING TEMPERATURES.  WIND CHILL
 40       FACTOR -17 DEGREES.  THURSDAY--CLEAR WITH MODERATING TEMPERATURES
 41       HIGH IN THE LOWER 30S.  WINDS 10 TO 15 MPH.  PROBABILITY OF
 42       PRECIPITATION NEAR 0 PERCENT THROUGH THURSDAY.
 43
 44
 45
 46
 47     111      116      107       40      124      105      115      120      105      122
 48     101      124      125      122      105      123       15       15       12      110
 49     111      107      110       40      111      116       40      124      110      105
 50      40      114      117      127      105      122       40       63       60      123
 51      56       40       40      127      111      116      104      123       40       61
 52      60       40      124      117       40       61       65       40      115      120
 53     110       56       40       40      120      122      117      102      101      102
 54     111      114      111      124      131       40      117      106       15       15
 55      12      120      122      105      103      111      120      111      124      101
 56     124      111      117      116       40      116      105      101      122       40
 57      60       40      120      105      122      103      105      116      124       40
 58     124      110      122      117      125      107      110       40      124      110
 59     125      122      123      104      101      131       56       15       15       12
 60       3       11       11       11       11       11       11       11       40       40
R
```

Figure 2. Sample run of PRTPRODUCT, showing original product and Dasher output.

```
      SUBROUTINE GETPARFD(HOLDER, LEVEL, PARAM, VALUE, ERROR)
C
C>>    THIS SUBROUTINE EXTRACTS TEMPERATURE, WIND DIRECTION OR
C      WIND SPEED FOR A GIVEN LEVEL FROM AN FD WINDS-ALOFT FORECAST.
C
C>>    USES SUBROUTINE NUMVALUE
C
C>>    POSSIBLE VALUES OF PARAM ARE:
C          WD     WIND DIRECTION
C          WS     WIND SPEED
C          TP     TEMPERATURE
C
      INTEGER LEVEL, PARAM, VALUE, HOLDER(512)
      INTEGER NBLANKS, FIELDSIZE, POINTER, OFFSET, CHRVALUE(3), CHRVALUE1(3),
     &  VALUE1, LOCAT
      REAL RVALUE, RVALUE1
      LOGICAL ERROR, ERROR1
C
      ERROR=.FALSE.
      ERROR1=.FALSE.
C
C>>    CHECK FOR UNAVAILABLE DATA
      IF(PARAM .EQ. "TP" .AND. LEVEL .EQ. 3)GO TO 2
C
C      FIND DATA GROUP FOR THE GIVEN LEVEL
      POINTER=0
      IF(LEVEL .EQ. 3)POINTER=28
      IF(LEVEL .EQ. 6)POINTER=33
      IF(LEVEL .EQ. 9)POINTER=41;       700 MB
      IF(LEVEL .EQ. 12)POINTER=49
      IF(LEVEL .EQ. 18)POINTER=57;      500 MB
      IF(LEVEL .EQ. 24)POINTER=65
      IF(LEVEL .EQ. 30)POINTER=73;      300 MB
      IF(LEVEL .EQ. 34)POINTER=80;      250 MB
      IF(LEVEL .EQ. 39)POINTER=89;      200 MB
      IF(POINTER .EQ. 0)GO TO 3
C
C      FIND OFFSET W/IN SELECTED DATA GROUP FOR GIVEN PARAM
      OFFSET=-1
      IF(PARAM .EQ. "WD")OFFSET=0
      IF(PARAM .EQ. "WS")OFFSET=2
      IF(PARAM .EQ. "TP")OFFSET=4
      IF(OFFSET .EQ. -1)GO TO 4
C
C      FIND FIELDSIZE FOR THE PARTICULAR PARAM
      FIELDSIZE=2
      IF(LEVEL .LT. 30 .AND. PARAM .EQ. "TP")FIELDSIZE=3
C
      LOCAT=POINTER+OFFSET
      DO 1 INDEX=1, FIELDSIZE
         CHRVALUE (INDEX)=HOLDER(LOCAT+(INDEX-1))
         IF(PARAM.EQ."WS")CHRVALUE1(INDEX)=HOLDER((LOCAT-2)+(INDEX-1))
1     CONTINUE
C
C>>    CHECK IF DATA IS AVAILBLE FOR THIS LEVEL
      IF(CHRVALUE(1).EQ.040K.AND.CHRVALUE(2).EQ.040K)GO TO 5
C
C      CONVERT EXTRACTED VALUE TO NUMERIC FORM AND RESTORE DIGITS
      CALL NUMVALUE(CHRVALUE, FIELDSIZE, RVALUE, ERROR1)
      IF(PARAM.EQ."WS")CALL NUMVALUE(CHRVALUE1,FIELDSIZE,RVALUE1,ERROR1)
      IF(ERROR1)GO TO 6
      VALUE=IFIX(RVALUE)
      VALUE1=IFIX(RVALUE1)
      IF(PARAM .EQ. "TP" .AND. LEVEL .GE. 30)VALUE=-VALUE
      IF(PARAM .EQ. "WD" .AND. VALUE .GT.36)VALUE=VALUE-50
      IF(PARAM .EQ. "WD")VALUE=VALUE*10
      IF(PARAM .EQ. "WS" .AND. VALUE1 .GT. 36)VALUE=VALUE+100
      GO TO 7
C
C     ERROR MESSAGES
2     TYPE "TEMPERATURE NA FOR 3000 FT LEVEL--GETPARFD"
      GO TO 6
3     TYPE "ILLEGAL LEVEL-LEGAL LEVELS ARE 3,6,9,12,18,24,30,34
     & AND 39 (THSND) FT--GETPARFD"
      GO TO 6
4     TYPE "ILLEGAL PARAM-LEGAL PARAMS ARE TP, WD AND WS--GETPARFD"
      GO TO 6
5     TYPE "NO DATA AVAILABLE FOR THIS LEVEL AT THIS STATION--GETPARFD"
6     ERROR=.TRUE.
7     RETURN
      END
      SUBROUTINE GETPARFRH(UPBUFFER, STATION, HOUR, PARAM, VALUE, ERROR)
C
C>>    GETPARFRH ACCESSES A COPY OF AN FRH HELD IN UPBUFFER AND RETURNS THE
C      VALUE OF A GIVEN PARAMETER (PARAM) FOR A GIVEN STATION AND TIME.
C
C>>    USES SUBROUTINES SKIPLINE2 AND NUMVALUE; UTIL.LB
C
C>>    POSSIBLE VALUES OF PARAM ARE:
C             RH                MEAN RELATIVE HUMIDITY, LOWEST 3 LAYERS
C             R1                RELATIVE HUMIDITY OF BOUNDARY LAYER
C             R2                RELATIVE HUMIDITY OF LOWEST TROP LAYER
C             R3                RELATIVE HUMIDITY OF MIDDLE TROP LAYER
C             VV                VERTICAL VELOCITY, MICROBAR PER SEC
C             LI                LIFTED INDEX
C             HH                1000-500 MB THICKNESS, DM
C             DD                WIND DIRECTION
C             FF                WIND SPEED
C             TB                BOUNDARY LAYER TEMP, DEG K
C             PS                BOUNDARY LAYER PRESSURE
C             PT                6 HR ACCUM PRECIP, HUNDRETHS OF INCH
C
C
      INTEGER STATION(2), HOUR, PARAM, UPBUFFER(5120)
      INTEGER LINENO, COLNO, SIZE, TARGET(3), POINTER, UPSTATION(4)
      REAL VALUE
      LOGICAL FLAG, ERROR1, ERROR
C
      ERROR1=.FALSE.
      ERROR=.FALSE.
C
C>>    CHECK FOR UNAVAILABLE DATA
      IF(HOUR .EQ. 0 .AND. (PARAM .EQ. "VV" .OR. PARAM .EQ. "PT"))GO TO 9
C
C>>    SET TARGET LOCATION AND SIZE ACCORDING TO HOUR PARAM AND SIZE
      LINENO=0
      DO 1 I=0, 48, 6
         IF(HOUR .EQ. I)LINENO=(I/6)+1
1     CONTINUE
      IF(LINENO .EQ. 0)GO TO 10
C
      COLNO=0
      IF(PARAM .EQ. "RH")COLNO=5
      IF(PARAM .EQ. "R1")COLNO=8
      IF(PARAM .EQ. "R2")COLNO=10
      IF(PARAM .EQ. "R3")COLNO=12
      IF(PARAM .EQ. "VV")COLNO=15
      IF(PARAM .EQ. "LI")COLNO=18
      IF(PARAM .EQ. "HH")COLNO=21
      IF(PARAM .EQ. "DD")COLNO=23
      IF(PARAM .EQ. "FF")COLNO=25
      IF(PARAM .EQ. "TB")COLNO=28
      IF(PARAM .EQ. "PS")COLNO=30
      IF(PARAM .EQ. "PT")COLNO=32
      IF(COLNO .EQ. 0)GO TO 11
C
      IF(PARAM .EQ. "VV" .OR. PARAM .EQ. "PT")GO TO 2
      SIZE=2
      GO TO 3
2     SIZE=3
3     CONTINUE
```

```
C
C>>     FIND STATION ID IN PRODUCT BODY
        CALL PUNPACK(STATION, 4, UPSTATION)
        DO 4 POINTER=1, 5118
           IF(UPSTATION(1) .EQ. UPBUFFER(POINTER) .AND. UPSTATION(2)
     &      .EQ. UPBUFFER(POINTER+1) .AND. UPSTATION(3) .EQ. UPBUFFER
     &       (POINTER+2))GO TO 5
4       CONTINUE
        GO TO 12
C
C>>     ONCE THE STATION IS FOUND, GO TO PROPER LINE AND COLUMN
5       FLAG=.FALSE.
        IF(UPBUFFER (POINTER-1) .EQ. 040K .AND. UPBUFFER(POINTER-2)
     &  .EQ. 057K)FLAG=.TRUE.
        IF(LINENO .EQ. 1)GO TO 6
        CALL SKIPLINE2(UPBUFFER, LINENO, POINTER)
6       POINTER=POINTER+(COLNO-1)
        IF(FLAG .AND. LINENO .NE. 1)POINTER=POINTER+35
C
C>>     EXTRACT NUMERICAL VALUE FROM PRODUCT BODY
        TARGET(1)=UPBUFFER(POINTER)
        TARGET(2)=UPBUFFER(POINTER+1)
        IF(SIZE .EQ. 3)TARGET(3)=UPBUFFER(POINTER+2)
        CALL NUMVALUE(TARGET, SIZE, VALUE, ERROR1)
        IF(ERROR1)GO TO 13
C
C>>     RESTORE IMPLIED DIGITS TO VALUE
        IF(PARAM .EQ. "VV")VALUE=VALUE/10
        IF(PARAM .EQ. "LI".AND. VALUE .GT. 50)VALUE=VALUE-100
        IF(PARAM .EQ. "DD")VALUE=VALUE*10
        IF(PARAM .EQ. "TB" .AND. VALUE .LT. 50)VALUE=VALUE+300
        IF(PARAM .EQ. "TB" .AND.(VALUE .GE. 50 .AND. VALUE .LE.99))
     &  VALUE=VALUE+200
        IF(PARAM.EQ. "PS" .AND. VALUE .LT. 60)VALUE=VALUE+1000
        IF(PARAM .EQ. "PS" .AND. (VALUE .GE. 60 .AND. VALUE .LE. 99))
     &  VALUE=VALUE+900
        IF(PARAM .EQ. "PT")VALUE=VALUE/100
C
C       SPECIAL HANDLING FOR THICKNESS
C       GET BOUNDARY LAYER TEMPERATURE
        IF(PARAM.NE."HH")GO TO 14
        POINTER=POINTER+7
        TARGET(1)=UPBUFFER(POINTER)
        TARGET(2)=UPBUFFER(POINTER+1)
        CALL NUMVALUE(TARGET,SIZE,VALUE2, ERROR1)
        IF(ERROR1)GO TO 13
        IF(VALUE2.LT.50)VALUE2=VALUE2+300
        IF(VALUE2.GE.50)VALUE2=VALUE2+200
        IF(VALUE2.LT.270.AND.VALUE.GE.60)GO TO 7
        IF(VALUE2.GT.295.AND.VALUE.LE.20)GO TO 8
        VALUE=VALUE+500
        GO TO 14
7       VALUE=VALUE+400
        GO TO 14
8       VALUE=VALUE+600
        GO TO 14
C
C>>     ERROR MESSAGES
9       TYPE "NO PREDICTED VALUE FOR THIS PARAMETER AT 0 HOURS-GETPARFRH"
        GO TO 13
```

```
10      TYPE "ERROR IN HOUR INPUT-GETPARFRH"
        GO TO 13
11      TYPE "ERROR IN PARAM INPUT-GETPARFRH"
        GO TO 13
12      TYPE "STATION CANNOT BE FOUND-GETPARFRH"
        GO TO 13
13      ERROR=.TRUE.
14      RETURN
        END
```

```
      SUBROUTINE GETPARFTJ(UPBUFFER, STATION, HOUR, LEVEL, PARAM, VALUE, ERROR)
C
C>>    GETPARFTJ ACCESSES A COPY OF AN FTJ HELD IN UPBUFFER AND RETURNS THE
C      VALUE OF A GIVEN PARAMETER (PARAM) FOR A GIVEN STATION, PRESSURE
C      LEVEL (EXCEPT FOR KI) AND TIME.
C
C>>    USES SUBROUTINES SKIPLINE2 AND NUMVALUE; UTIL.LB
C
C>>    POSSIBLE VALUES OF PARAM ARE:
C             LA                LATITUDE (DEGREES AND 10THS)
C             LO                LONGITUDE (DEGREES AND 10THS)
C             PP                PRESSURE (MB)
C             MP                MODEL PRESSURE (MB)
C             TT                TEMPERATURE (CELSIUS)
C             TD                DEWPOINT (CELSIUS)
C             KI                K INDEX
C
C
      COMMON/FTJ2/IRMHIGH(20)
      INTEGER UPBUFFER(5120), STATION(2), NSTATION, HOUR, LEVEL, PARAM
      INTEGER LINENO, COLNO, SIZE, INDEX, POINTER, TARGET(5), UPSTATION(4)
      REAL VALUE
      LOGICAL ERROR, ERROR1, HIALT
C
      DATA IRMHIGH/"DEN LND BOI ELP ABQ UCC ALS RNO SLC PIH "/
C
C>>    CHECK FOR STATIONS MISSING 850MB VALUES
      HIALT=.FALSE.
      DO 1 INDEX=1,19,2
         IF(STATION(1).EQ.IRMHIGH(INDEX).AND.STATION(2).EQ.IRMHIGH
     &     (INDEX+1))HIALT=.TRUE.
1     CONTINUE
C
C>>    SET TARGET LOCATION  ACCORDING TO LEVEL, PARAM AND HOUR
      LINENO=0
      COLNO=0
      ERROR=.FALSE.
      ERROR1=.FALSE.
      IF(LEVEL .EQ. 700)LINENO=1
      IF(LEVEL .EQ. 850)LINENO=2
      IF(LEVEL .EQ. 0)LINENO=3
      IF(HIALT .AND. LEVEL .EQ. 850)GO TO 13
      IF(HIALT .AND. LEVEL .EQ. 0)LINENO=2
      IF(PARAM .EQ. "KI")LINENO=1
      IF(PARAM .EQ. "MP")LINENO=3
      IF(HIALT .AND. PARAM .EQ. "MP")LINENO=2
      IF(LINENO .EQ. 0)GO TO 14
C
      IF(PARAM .EQ. "LA")GO TO 2
      IF(PARAM .EQ. "LO")GO TO 3
      IF(PARAM .EQ. "PP")GO TO 4
      IF(PARAM .EQ. "TT")GO TO 5
      IF(PARAM .EQ. "TD")GO TO 6
      IF(PARAM .EQ. "KI")GO TO 7
      IF(PARAM .EQ. "MP")GO TO 8
      GO TO 15
C
2     IF(HOUR .EQ. 00)COLNO=10
      IF(HOUR .EQ. 06)COLNO=20
      IF(HOUR .EQ. 12)COLNO=30
```

```
        IF(HOUR .EQ. 18)COLNO=40
        IF(HOUR .EQ. 24)GO TO 16
        GO TO 9
 3      IF(HOUR .EQ. 00)COLNO=13
        IF(HOUR .EQ. 06)COLNO=23
        IF(HOUR .EQ. 12)COLNO=33
        IF(HOUR .EQ. 18)COLNO=43
        IF(HOUR .EQ. 24)GO TO 16
        GO TO 9
 4      IF(HOUR .EQ. 00)COLNO=16
        IF(HOUR .EQ. 06)COLNO=26
        IF(HOUR .EQ. 12)COLNO=36
        IF(HOUR .EQ. 18)COLNO=46
        IF(HOUR .EQ. 24)GO TO 16
        GO TO 9
 5      IF(HOUR .EQ. 0 .OR. HOUR .EQ. 6 .OR. HOUR .EQ. 12 .OR. HOUR .EQ. 18)GO TO 17
        IF(HOUR .EQ. 24)COLNO=53
        GO TO 9
 6      IF(HOUR .EQ. 00 .OR. HOUR .EQ. 6 .OR. HOUR .EQ. 12 .OR. HOUR .EQ. 18)GO TO 17
        IF(HOUR .EQ. 24)COLNO=59
        GO TO 9
 7      IF(HOUR .EQ. 0 .OR. HOUR .EQ. 6 .OR. HOUR .EQ. 12 .OR. HOUR .EQ. 18)GO TO 17
        IF(HOUR .EQ. 24)COLNO=65
        GO TO 9
 8      COLNO=50
C
 9      IF(COLNO .EQ. 0)GO TO 18
        SIZE=3
        IF(PARAM .EQ. "TT" .OR. PARAM .EQ. "TD")SIZE=5
C
C>>     FIND STATION NAME IN PRODUCT BODY
        CALL PUNPACK(STATION, 4, UPSTATION)
           DO 10 POINTER=1, 5118
              IF(UPSTATION(1) .EQ. UPBUFFER(POINTER) .AND. UPSTATION (2)
     &         .EQ. UPBUFFER(POINTER+1) .AND. UPSTATION(3) .EQ. UPBUFFER
     &         (POINTER+2))GO TO 11
10         CONTINUE
        GO TO 19
C
C>>     ONCE THE STATION IS FOUND, GO TO THE PROPER LINE
11      CALL SKIPLINE2(UPBUFFER, LINENO, POINTER)
C
C>>     ONCE ON THE PROPER LINE, GO TO THE PROPER COLUMN
        IF(LINENO .EQ. 1)COLNO=COLNO-4
        POINTER=POINTER+(COLNO-1)
        DO 12 INDEX=1, SIZE
           TARGET(INDEX)=UPBUFFER(POINTER+INDEX-1)
12      CONTINUE
C
C>>     TRANSLATE CHARACTERS IN TARGET INTO NUMERICAL VALUE
        CALL NUMVALUE(TARGET, SIZE, VALUE, ERROR1)
        IF(ERROR1)GO TO 20
C
C>>     RESTORE IMPLIED DIGITS
        IF(PARAM .EQ. "LO" .OR. PARAM .EQ. "LA")VALUE=VALUE/10
        IF(PARAM .EQ. "LO" .AND. VALUE .LT. 50)VALUE=VALUE+100
        IF(PARAM .NE. "PP" .AND. PARAM .NE. "MP")GO TO 21
        IF(VALUE .LT. 100)VALUE=VALUE+1000
        GO TO 21
C
```

```
C>>     ERROR MESSAGES
13      TYPE "HIGH ALTITUDE STATION, 850MB DATA NT AVLBL--GETPARFTJ"
        GO TO 20
14      TYPE "ILLEGAL LEVEL-LEGAL LEVELS ARE 0, 700, 850--GETPARFTJ"
        GO TO 20
15      TYPE "ILLEGAL PARAMETER-LEGAL PARAMS ARE LA, LO, PP, TT, TD, KI AND MP-GETPARFTJ"
        GO TO 20
16      TYPE "24-HOUR FCST NT AVBL FOR LA, LO AND PP--GETPARFTJ"
        GO TO 20
17      TYPE "ONLY 24-HOUR FCSTS AVBL FOR TT, TD AND KI--GETPARFTJ"
        GO TO 20
18      TYPE "ILLEGAL HOUR-LEGAL VALUES ARE 0, 6, 12, 18 AND 24--GETPARFTJ"
        GO TO 20
19      TYPE "STATION NOT FOUND--GETPARFTJ"
20      ERROR=.TRUE.
21      RETURN
        END
```

```
      SUBROUTINE GETPRODUCT(KEY, HOLDER, EXTIME, ERROR)
C
C>>    GETPRODUCT WILL EXTRACT A PRODUCT (DESIGNATED BY ARRAY KEY) FROM
C      THE AFOS DATA BASE AND PLACE IT IN AN UNPACKED ARRAY (HOLDER).
C      THE JULIAN TIME OF THE PRODUCT CAN BE CHECKED AGAINST THE CURRENT
C      JULIAN TIME.  IF THE DIFFERENCE IS GREATER THAN THE GIVEN
C      EXPIRATION INTERVAL (EXTIME, IN MINUTES), AN ERROR MESSAGE
C      IS RETURNED.  IF EXTIME IS NEGATIVE, NO TIME CHECK IS MADE.
C      THE PRODUCT CAN BE UP TO 15 AFOS BLOCKS LONG.
C
C>>    USES SUBROUTINES CURJTIME AND KEYJTIME BY JACK MAY; BG.LB, UTIL.LB
C
      INTEGER KEY(5), KEYREC(15), PACBUF(128), UNPACBUF(256),
     & HOLDER(7680), TERM, START, PLACE, NBLKS, COUNTER, TODAY(3),
     & NOW(3), EXTIME
      REAL CURJT, PRODJT, TDIFF
      LOGICAL ERROR
      NBLKS=0
      ERROR=.FALSE.
C
C>>    SEARCH FOR KEY RECORD
      CALL KSRCF(KEY, KEYREC, IER)
      IF(IER .NE. 1)GO TO 5
      IF(EXTIME .LT. 0)GO TO 1
C
C>>    CHECK TIME OF PRODUCT
      CALL DATE(TODAY, IER)
      CALL TIME(NOW, IER)
      CALL CURJTIME(TODAY, NOW(1), NOW(2), NOW(3), CURJT)
      CALL KEYJTIME(KEY, PRODJT, IER)
      TDIFF=CURJT-PRODJT
      IF(TDIFF.GT.EXTIME)GO TO 6
C
C>>    READ THE FIRST BLOCK
1     CALL RDBKF(0, PACBUF, IER)
      IF(IER .NE. 1)GO TO 5
C
C>>    UNPACK THE FIRST BLOCK AND READ IT BYTE BY BYTE INTO
C      THE ARRAY HOLDER (SKIP 23 BYTES OF BLOCK HEADER)
      CALL PUNPACK(PACBUF, 256, UNPACBUF)
      TERM=233
      DO 2 COUNTER=1, TERM
         HOLDER(COUNTER)=UNPACBUF(COUNTER+23)
2     CONTINUE
      NBLKS=NBLKS+1
C
C>>    IF THERE ARE MORE BLOCKS OF THE PRODUCT, READ ANOTHER BLOCK,
C      UNPACK IT, AND PLACE IT IN HOLDER (SKIP 4 BYTES OF BLOCK HEADER).
3     IF(PACBUF(1) .EQ. -1) GO TO 8
      CALL NXBKF(PACBUF, IER)
      CALL PUNPACK(PACBUF, 256, UNPACBUF)
      START=TERM+1
      TERM=TERM+252
      PLACE=5
      DO 4 COUNTER=START, TERM
         HOLDER(COUNTER)=UNPACBUF(PLACE)
         PLACE=PLACE+1
4     CONTINUE
      NBLKS=NBLKS+1
C     TEST FOR MAXIMUM NUMBER OF BLOCKS
```

```
       IF(NBLKS .GT. 15)GO TO 7
       GO TO 3
C
C>>     ERROR AND STATUS MESSAGES
5       TYPE"PRODUCT NOT STORED"
        ERROR=.TRUE.
        GO TO 10
6       TYPE "PRODUCT TOO OLD"
        ERROR=.TRUE.
        GO TO 10
7        TYPE "PRODUCT LARGER THAN FIFTEEN BLOCKS"
        ERROR=.TRUE.
8       CONTINUE
X       WRITE(10, 9)NBLKS
X9      FORMAT(1X, I2, " AFOS BLOCKS IN UPACKED ARRAY")
10      RETURN
        END



        SUBROUTINE KEYJTIME(KEY,PRODJT,IER)
C
C THIS SUBROUTINE RETRIEVES THE JULIAN TIME (MINUTES SINCE MIDNIGHT
C JANUARY 1ST) FROM DATAKEY0 OF THE PRODUCT DEFINED IN KEY.
C
C JACK MAY/ WSFO CLEVELAND/ FTS 293-4949
C
        DIMENSION KEY(5),KREC(20)
        INTEGER UNKREC(40),A0,A1,A2
C
C PUT KEY RECORD INTO VARIABLE ARRAY KREC AND UNPACK INTO ARRAY UNKREC
        CALL KSRCF  (KEY,KREC,IER)
          IF (IER.NE.1) GOTO 900
        CALL UNPACK (KREC,40,UNKREC)
C
C JULIAN TIME NOW CONTAINED IN THREE WORDS OF UNKREC (WORDS 19,20,21).
C BELOW THEY ARE DEFINED AS A0, A1, AND A2.
C
C JULIAN TIME = A0(2**14) + A1(2**7) + A2
C
        A0 = UNKREC(19)
        A1 = UNKREC(20)
        A2 = UNKREC(21)
C
        PRODJT = 0.
        XNUM1  = A0 * (2.**14)
        XNUM2  = A1 * (2.**7 )
        XNUM3  = A2
        PRODJT = XNUM1 + XNUM2 + XNUM3
C
        IER    = 1
        RETURN
  900   IER    = 0
        RETURN
        END
```

```
      SUBROUTINE CURJTIME(NDATE,IHOUR,MIN,ISEC,CURJT)
C CURJTIME FIGURES THE CURRENT JULIAN MINUTE FROM THE STATION CLOCK.
C "CURJT" IS THE VARIABLE PASSED BACK TO THE PROGRAM.
C
      DIMENSION NDATE(3)
C
      MONTH = NDATE(1)
      IDATE = NDATE(2)
      IYEAR = NDATE(3)
C
C .. FIGURE IF LEAP YEAR
      LYEAR = IYEAR-(4*(IFIX(IYEAR/4.)))      ; IF LYEAR = 0, LEAPYEAR
C
      CURJT = 0
      IF(MONTH.GE. 2) CURJT = CURJT + 44640.  ; ADD JAN MINUTES
      IF(MONTH.GE. 3) CURJT = CURJT + 40320.  ; ADD FEB MINUTES
      IF(LYEAR.EQ. 0) CURJT = CURJT +  1440.  ; ADD LEAP YEAR MINUTES
      IF(MONTH.GE. 4) CURJT = CURJT + 44640.  ; ADD MAR MINUTES
      IF(MONTH.GE. 5) CURJT = CURJT + 43200.  ; ADD APR MINUTES
      IF(MONTH.GE. 6) CURJT = CURJT + 44640.  ; ADD MAY MINUTES
      IF(MONTH.GE. 7) CURJT = CURJT + 43200.  ; ADD JUN MINUTES
      IF(MONTH.GE. 8) CURJT = CURJT + 44640.  ; ADD JUL MINUTES
      IF(MONTH.GE. 9) CURJT = CURJT + 44640.  ; ADD AUG MINUTES
      IF(MONTH.GE.10) CURJT = CURJT + 43200.  ; ADD SEP MINUTES
      IF(MONTH.GE.11) CURJT = CURJT + 44640.  ; ADD OCT MINUTES
      IF(MONTH.GE.12) CURJT = CURJT + 43200.  ; ADD NOV MINUTES
C
C
      CURJT  = CURJT  + (IDATE-1)   *1440.    ; ADD DAYS SINCE LAST MONTH
      CURJT  = CURJT  + (IHOUR*60.)           ; ADD NUMBER OF HOURS PAST MIDN
      CURJT  = CURJT  + FLOAT(MIN)            ; ADD NUMBER OF MINUTES
C
      RETURN
      END
      SUBROUTINE EXTRFRH(INSTATS, OUTSTATS, NSTATS, HOUR, PARAM, RESULTS, ERROR)
C
C
C>>    EXTRFRH TAKES A LIST OF STATIONS, DIVIDES IT INTO GROUPS
C      CORRESPONDING TO THE DIFFERENT FRH BULLETINS, CALLS UP THE
C      BULLETINS FROM THE DATA BASE AND EXTRACTS THE DATA FOR EACH
C      DESIRED STATION FOR THE GIVEN HOUR AND PARAMETER.  THIS
C      SUBROUTINE CAN HANDLE BULLETINS 60 THROUGH 73.
C
C>>    USES SUBROUTINES MATCH, GETPRODUCT AND GETPARFRH; UTIL.LB, TOP.LB
C
C
      COMMON/FRH/IFRH6073(14,6,2), IXXLIST(14)
C
C     PASSED VARIABLES
      INTEGER INSTATS(NSTATS,2), OUTSTATS(NSTATS,2), NSTATS, HOUR, PARAM
      REAL RESULT6 (NSTATS)
C
C     LOCAL VARIABLES
      INTEGER HOLDER(5120), REFLIST(6,2), STATLIST(6,2), NUMLIST, TOTAL,
     & STAT(2), UPKEY(10), KEY(5), KEYTEMP(2), INDEX, WINDEX, ZINDEX, PTR,
     & EXPIRE, NREF
      REAL VALUE
      LOGICAL ERROR1, ERROR2, ERROR
C
C
C>>    FILL THE MATRIX IFRH6073 WITH ALL THE STATION ID'S IN
C      THE FRH BULLETINS (14 BULLETINS WITH 6 2-WORD ID'S EACH).
C      THE ID'S ARE SCRAMBLED BECAUSE DG'S FORTRAN IV DOES NOT PERMIT
C      LOOPING IN DATA STATEMENTS TO CONTROL THE WAY THE MATRIX IS
C      LOADED.
C
      DATA IFRH6073/"PWALDCCABUSTBHDEDSOKBILBSESF"/
      DATA IFRH6073(1,2,1)/"CABTORSAPISDMOSSTODFFSELGEFA"/
      DATA IFRH6073(1,3,1)/"BGBORDMICLMEJAMKDDSARAABPDLA"/
      DATA IFRH6073(1,4,1)/"COLGHALADABNNEMSOMHOGTPHMFRN"/
      DATA IFRH6073(1,5,1)/"AFPHILTLCRTYSHINLBBRBIDEBOSL"/
      DATA IFRH6073(1,6,1)/"9BIPC73JINATLIORBFDRMSCYPICD"/
      DATA IFRH6073(1,1,2)/"M B A E F L M T M C S B A O "/
      DATA IFRH6073(1,2,2)/"R V F V T F B M P W D P G T "/
      DATA IFRH6073(1,3,2)/"R S U A E M N E C T P Q X X "/
      DATA IFRH6073(1,4,2)/"N A T L Y A W P A U F X R O "/
      DATA IFRH6073(1,5,2)/"A L M H W S V L F O L N I C "/
      DATA IFRH6073(1,6,2)/"6 T H 2 D L T D F T O S H C "/
C
C>>    FILL IXXLIST WITH KEY ENDINGS
      DATA IXXLIST/"6061626364656667686970717273"/
C
      ERROR1=.FALSE.
      ERROR2=.FALSE.
      EXPIRE=720
      NREF=6
      TOTAL=0
      UPKEY(1)="F"
      UPKEY(2)="R"
      UPKEY(3)="H"
      UPKEY(6)=0
      DO 1 INDEX=1,3
         UPKEY(INDEX)=ISHFT(UPKEY(INDEX), -8)
1     CONTINUE
```

```
C
C
C>>    PROCESS THE LIST OF STATIONS
C
       DO 10 INDEX=1,14
C
C>>       FIRST DETERMINE IF THERE ARE ANY STATIONS IN THE LIST FOR EACH
C         BULLETIN
C
          DO 3 WINDEX=1,6
             DO 2 ZINDEX=1,2
                REFLIST(WINDEX, ZINDEX)=IFRH6073(INDEX, WINDEX, ZINDEX)
2            CONTINUE
3         CONTINUE
C
          NUMLIST=0
          CALL MATCH(INSTATS, NSTATS, REFLIST, NREF, STATLIST, NUMLIST)
C
C>>       THEN, IF THERE ARE ANY STATIONS LISTED FOR THE BULLETIN, CREATE
C         THE PROPER KEY AND GET THE DATA
C
          IF(NUMLIST .LT. 1)GO TO 7
C
C         CREATE KEY
          CALL PUNPACK(IXXLIST(INDEX), 2, KEYTEMP)
          UPKEY(4)=KEYTEMP(1)
          UPKEY(5)=KEYTEMP(2)
          CALL PACK(UPKEY, 9, KEY)
          CALL KFILL(KEY, IER)
C
C         RETREIVE PRODUCT
          CALL GETPRODUCT(KEY, HOLDER, EXPIRE, ERROR1)
          IF(ERROR1)GO TO 8
          DO 7 WINDEX=1,NUMLIST
             STAT(1)=STATLIST(WINDEX,1)
             STAT(2)=STATLIST(WINDEX,2)
             CALL GETPARFRH(HOLDER, STAT, HOUR, PARAM, VALUE, ERROR2)
             IF(ERROR2)GO TO 5
             OUTSTATS(TOTAL+WINDEX,1)=STAT(1)
             OUTSTATS(TOTAL+WINDEX,2)=STAT(2)
             RESULTS(TOTAL+WINDEX)=VALUE
             GO TO 7
5            WRITE(10, 6)(STAT(PTR),PTR=1,2)
6            FORMAT(1X, "DATA FOR ", 2A2, "IS NOT AVAILABLE")
7         CONTINUE
C
          TOTAL=TOTAL+NUMLIST
C
          GO TO 10
8         WRITE(10,9)(KEY(PTR),PTR=1,5)
9         FORMAT(1X, "DATA FROM ", 5A2, "IS NOT AVAILABLE")
C
10     CONTINUE
       IF(ERROR1.OR.ERROR2)ERROR=.TRUE.
       RETURN
       END
```

```
      SUBROUTINE EXTRFTJ(INSTATS,OUTSTATS,NSTATS,HOUR,LEVEL,PARAM,RESULTS,ERROR)
C
C
C>>    EXTRFTJ TAKES A LIST OF STATIONS, DIVIDES IT INTO GROUPS
C      CORRESPONDING TO THE DIFFERENT FTJ BULLETINS, CALLS UP THE
C      BULLETINS FROM THE DATA BASE AND EXTRACTS THE DATA FOR
C      EACH DESIRED STATION FOR THE GIVEN HOUR, LEVEL AND PARAMETER.
C      THIS SUBROUTINE CAN HANDLE BULLETINS 50 THROUGH 57.
C
C>>    USES SUBROUTINES MATCH, GETRPRODUCT, AND GETPARFTJ; UTIL.LB, TOP.LB
C
      COMMON/FTJ/IFTJ5057(8, 10, 2), IXXLIST(8)
C
C     PASSED VARIABLES
      INTEGER INSTATS(NSTATS, 2), OUTSTATS(NSTATS, 2), NSTATS, HOUR,
     &  PARAM, LEVEL
      REAL RESULTS(NSTATS)
C
C     LOCAL VARIABLES
      INTEGER HOLDER(7680), REFLIST(10, 2), STATLIST(10, 2), NUMLIST,
     &  TOTAL, STAT(2), UPKEY(10), KEY(5), KEYTEMP(2), INDEX, WINDEX,
     &  ZINDEX, PTR, EXPIRE
      REAL VALUE
      LOGICAL ERROR1, ERROR2, ERROR
C
C
C>>    FILL THE MATRIX IFTJ5057 WITH ALL THE STATION ID'S IN THE FTJ
C      BULLETINS (8 BULLETINS WITH A MAXIMUM OF 10 2-WORD ID'S EACH).  THE
C      ID'S ARE SCRAMBLED BECAUSE DG'S FORTRAN IV DOES NOT PERMIT LOOPING
C      IN DATA STATEMENTS TO CONTROL THE WAY THE MATRIX IS LOADED.
C
      DATA IFTJ5057/"SEBIGRCASFALTOLO"/
      DATA IFTJ5057(1,2,1)/"GELNSSPWRNDDUMTY"/
      DATA IFTJ5057(1,3,1)/"YKBIPIBTUCABLIGS"/
      DATA IFTJ5057(1,4,1)/"GTRAFNCOLAOKHOIL"/
      DATA IFTJ5057(1,5,1)/"BOLBINBOSALBMSCA"/
      DATA IFTJ5057(1,6,1)/"PDINCLALSLELMEAT"/
      DATA IFTJ5057(1,7,1)/"MFMSCRLGPHFTSTTL"/
      DATA IFTJ5057(1,8,1)/"@@DSPIBUDESAJALA"/
      DATA IFTJ5057(1,9,1)/"@@@@@@IP@@BRBHMI"/
      DATA IFTJ5057(1,10,1)/"@@@@@@DC@@@@@@@@"/
      DATA IFTJ5057(1,1,2)/"A L B R O S P U "/
      DATA IFTJ5057(1,2,2)/"G D M M O C N S "/
      DATA IFTJ5057(1,3,2)/"M S A V C Q T O "/
      DATA IFTJ5057(1,4,2)/"F P T N X C C M "/
      DATA IFTJ5057(1,5,2)/"I F D S N B Y E "/
      DATA IFTJ5057(1,6,2)/"X L E B C P M L "/
      DATA IFTJ5057(1,7,2)/"R P W A X W L H "/
      DATA IFTJ5057(1,8,2)/"@ M T F N T N L "/
      DATA IFTJ5057(1,9,2)/"@ @ @ T @ O M A "/
      DATA IFTJ5057(1,10,2)/"@ @ @ A @ @ @ @ "/
C
C>>    FILL IXXLIST WITH KEY ENDINGS
      DATA IXXLIST/"5051525354555657"/
C
      ERROR1=.FALSE.
      ERROR2=.FALSE.
      EXPIRE=720
      TOTAL=0
      UPKEY(1)="F"
```

```
        UPKEY(2)="T"
        UPKEY(3)="J"
        UPKEY(6)=0
        DO 1 INDEX=1,3
           UPKEY(INDEX)=ISHFT(UPKEY(INDEX), -8)
 1      CONTINUE
 C
 C
 C>>    PROCESS THE LIST OF STATIONS
 C
        DO 10 INDEX=1,8
 C
 C>>       FIRST DETERMINE IF THERE ARE ANY STATIONS IN THE LIST FOR EACH
 C         BULLETIN
 C
           DO 3 WINDEX=1, 10
              DO 2 ZINDEX=1,2
                 REFLIST(WINDEX, ZINDEX)=IFTJ5057(INDEX, WINDEX, ZINDEX)
 2            CONTINUE
 3         CONTINUE
 C
           NUMLIST=0
           CALL MATCH(INSTATS,NSTATS,REFLIST,10,STATLIST,NUMLIST)
 C
 C>>       THEN, IF THERE ARE ANY STATIONS LISTED FOR THE BULLETIN,
 C         CREATE THE PROPER KEY AND GET THE DATA
 C
           IF(NUMLIST .LT. 1)GO TO 7
 C
 C         CREATE KEY
           CALL PUNPACK(IXXLIST(INDEX), 2, KEYTEMP)
           UPKEY(4)=KEYTEMP(1)
           UPKEY(5)=KEYTEMP(2)
           CALL PACK(UPKEY, 9, KEY)
           CALL KFILL(KEY, IER)
 C
 C         RETREIVE PRODUCT
           CALL GETPRODUCT(KEY, HOLDER, EXPIRE, ERROR1)
           IF(ERROR1)GO TO 8
           DO 7 WINDEX=1, NUMLIST
              STAT(1)=STATLIST(WINDEX, 1)
              STAT(2)=STATLIST(WINDEX, 2)
              CALL GETPARFTJ(HOLDER, STAT, HOUR, LEVEL, PARAM, VALUE, ERROR2)
              IF(ERROR2)GO TO 5
              OUTSTATS(TOTAL+WINDEX, 1)=STAT(1)
              OUTSTATS(TOTAL+WINDEX, 2)=STAT(2)
              RESULTS(TOTAL+WINDEX)=VALUE
              GO TO 7
 5            WRITE(10, 6)(STAT(PTR),PTR=1,2)
 6            FORMAT(1X, "DATA FOR ",2A2, "IS NOT AVAILABLE")
 7         CONTINUE
 C
           TOTAL=TOTAL+NUMLIST
 C
           GO TO 10
 8         WRITE(10, 9)(KEY(PTR), PTR=1,5)
 9         FORMAT(1X, "DATA FROM ", 5A2, "IS NOT AVAILABLE")
 C
 10     CONTINUE
        IF(ERROR1.OR.ERROR2)ERROR=.TRUE.
        RETURN
        END
```

```
      SUBROUTINE MATCH(INPUTLIST, NIN, REFLIST, NREF, MATCHLIST, MATCHNUM)
C
C     MATCH WILL TAKE AN INPUT LIST OF STATION ID'S AND COMPARE
C     IT TO A REFERENCE LIST OF STATION ID'S, PRODUCING A LIST
C     OF THE STATIONS THAT MATCH AND HOW MANY MATCHES WERE FOUND.
C     ALL ARRAYS ARE TWO-DIMENSIONAL.  ARRAY SIZE IS FLEXIBLE,
C     DEPENDING ON INPUT SIZES.
C
C     INPUT:  INPUTLIST...ARRAY OF STATION ID'S
C             NIN...ARRAY SIZE OF INPUTLIST
C             REFLIST...REFERENCE ARRAY OF STATION ID'S
C             NREF...ARRAY SIZE OF REFLIST
C
C     OUTPUT:  MATCHLIST...ARRAY OF MATCHING STATIONS
C              MATCHNUM...NUMBER OF MATCHING STATIONS
C
      INTEGER NIN, NREF, INPUTLIST(NIN,2), REFLIST(NREF,2),
     & MATCHLIST(NREF,2), INDEX, POINTER, LOCAT, MATCHNUM
      MATCHNUM=0
      LOCAT=1
      DO 2 INDEX=1, NIN
         DO 1 POINTER=1, NREF
            IF((INPUTLIST(INDEX,1) .EQ. REFLIST(POINTER,1)) .AND.
     &         (INPUTLIST(INDEX,2) .EQ. REFLIST(POINTER,2)))GO TO 3
            GO TO 1
3           MATCHLIST(LOCAT,1)=INPUTLIST(INDEX,1)
            MATCHLIST(LOCAT,2)=INPUTLIST(INDEX,2)
            MATCHNUM=MATCHNUM+1
            LOCAT=LOCAT+1
            GO TO 2
1        CONTINUE
2     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SKIPLINE2(UPBUFFER, LINENO, POINTER)
C
C>>   SKIPLINE2 WILL MOVE A POINTER DOWN A SPECIFIED NUMBER OF LINES
C     IN AN AFOS PRODUCT THAT HAS BEEN PLACED IN AN UNPACKED ARRAY.
C
      INTEGER UPBUFFER(5120), LINENO, POINTER, NLINES
C
      NLINES=LINENO-1
1     IF(UPBUFFER(POINTER) .EQ. 15K .AND. UPBUFFER(POINTER+1)
     & .EQ. 15K .AND. UPBUFFER(POINTER+2) .EQ. 12K)NLINES=NLINES-1
      IF(NLINES .EQ. 0)GO TO 2
      POINTER=POINTER+1
      GO TO 1
2     POINTER=POINTER+3
      RETURN
      END
      SUBROUTINE NUMVALUE (CHARS, FIELDSIZE, VALUE, ERROR)
C
C>>   CONVERTS ASCII CHARACTERS TO NUMERICAL VALUE.
C     OUTPUT IS REAL, LEADING BLANKS ARE IGNORED.
C
C>>   ADAPTED FROM FLTCVT BY MATT PEROUTKA
C
      INTEGER FIELDSIZE, CHARS(FIELDSIZE), INDEX, DIV, START, BREAK
      REAL WVALUE, DVALUE, VALUE
      LOGICAL NEGATIVE, DECIMAL, ERROR
C
      WVALUE=0.0
      DVALUE=0.0
      VALUE=0.0
      ERROR=.FALSE.
      NEGATIVE=.FALSE.
      DECIMAL=.FALSE.
      DIV=10
C
C>>   CYCLE PAST LEADING BLANKS, IF ANY
      INDEX=1
1     IF(CHARS(INDEX).NE.40K)GO TO 2
      IF(INDEX.GE.FIELDSIZE)GO TO 10
      INDEX=INDEX+1
      GO TO 1
C
C>>   CHECK SIGN
2     START=INDEX
      IF(CHARS(START).EQ.53K)START=START+1    ;+ SIGN
      IF(CHARS(START).NE.55K)GO TO 3
      NEGATIVE=.TRUE.
      START=START+1                           ;- SIGN
C
C>>   SEARCH FOR DECIMAL POINT
3     DO 4 INDEX=START, FIELDSIZE
          BREAK=INDEX
         IF(CHARS(INDEX).EQ.56K)GO TO 5
4     CONTINUE
      GO TO 6
5     DECIMAL=.TRUE.
C
C>>   PROCESS WHOLE PART (ALL IF NOT(DECIMAL))
6     LAST=FIELDSIZE
      IF(DECIMAL)LAST=BREAK-1
      DO 7 INDEX=START,LAST
         IF(CHARS(INDEX).LT.60K .OR. CHARS(INDEX).GT.71K)GO TO 11
         WVALUE=WVALUE*10+(CHARS(INDEX)-60K)
7     CONTINUE
C
C>>   PROCESS DECIMAL PART
      IF(NOT(DECIMAL))GO TO 9
      START=BREAK+1
      DO 8 INDEX=START,FIELDSIZE
         IF(CHARS(INDEX).LT.60K .OR. CHARS(INDEX).GT.71K)GO TO 11
         DVALUE=DVALUE+FLOAT((CHARS(INDEX)-60K))/DIV
         DIV=DIV*10
8     CONTINUE
C
C>>   SUM UP VALUES
9     VALUE=WVALUE
```

```
      IF(DECIMAL)VALUE=VALUE+DVALUE
      IF(NEGATIVE)VALUE=VALUE*(-1)
      GO TO 13
C
C>>   ERROR MESSAGES
10    TYPE "FIELD IS ALL BLANKS--NUMVALUE"
      GO TO 12
11    TYPE "ILLEGAL CHARACTER IN FIELD-NUMVALUE"
12    ERROR=.TRUE.
13    RETURN
      END
```

```
C       THIS PROGRAM WILL TYPE OUT AN AFOS PRODUCT (PUT INTO UNPACKED
C       ARRAY FORM BY SUBROUTINE GETPRODUCT) ON THE DASHER.
C       NO TIME CHECK IS MADE ON THE PRODUCT, WHICH CAN BE UP TO 15 AFOS BLOCKS LONG.
C
C       LOAD LINE:  RLDR PRTPRODUCT GETPRODUCT CURJTIME KEYJTIME BG.LB
C       UTIL.LB FORT.LB
C
        INTEGER HOLDER(5120)
        INTEGER ROW, COLUMN, INDEX, WINDEX, START, FINISH, NROWS
        INTEGER LEFTMASK, NULL, KEY(5), LAST, NOTIME
        LOGICAL ERROR
C
        LEFTMASK=177400K
        NULL=000K
        ÑOTIME=-1
C
C       OBTAIN PRODUCT KEY FROM DASHER
        TYPE "TYPE THE KEY OF PRODUCT YOU WISH TO PRINT OUT"
        READ(11,1)(KEY(INDEX),INDEX=1,5)
1       FORMAT(5A2)
        KEY(5)=IAND(KEY(5), LEFTMASK)
        KEY(5)=IOR(KEY(5), NULL)
C
        CALL GETPRODUCT(KEY, HOLDER, NOTIME, ERROR)
        IF(ERROR)GO TO 7
C
C       PRINT HEADING OF COLUMN NUMBERS
        WRITE(10,2)
2       FORMAT(1X, T9, "0", T16, "1", T23, "2", T30, "3",
      & T37, "4", T44, "5", T51, "6", T58, "7", T65, "8",
      & T72, "9")
C
C       DETERMINE LENGTH OF ACTUAL PRODUCT
        DO 3 INDEX=1, 5120
           LAST=INDEX
           IF(HOLDER(INDEX) .EQ. 003K)GO TO 4
3       CONTINUE
4       NROWS=LAST/10
        IF(MOD(LAST,10) .NE. 0)NROWS=NROWS+1
C
C       TYPE OUT PRODUCT, 10 WORDS PER ROW (LEADING ZEROES OMITTED)
        START=1
        DO 6 INDEX=1, NROWS
           ROW=INDEX-1
           FINISH=START+9
           WRITE(10,5)ROW, (HOLDER(WINDEX), WINDEX=START,FINISH)
5          FORMAT(1X, I3, 10(1X,O16))
           START=FINISH+1
6       CONTINUE
7       STOP
        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 (D822)
ENVIRONMENTAL DATA AND INFORMATION SERVICE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
U.S. DEPARTMENT OF COMMERCE

6009 Executive Boulevard
Rockville, MD 20852