U.S. DEPARTMENT OF COMMERCE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
NATIONAL WEATHER SERVICE
OFFICE OF SYSTEMS DEVELOPMENT
TECHNIQUES DEVELOPMENT LABORATORY

TDL OFFICE NOTE 93-2

AN ANALYSIS OF SOME FEATURES OF THE GRIB CODE

Harry R. Glahn

May 1993

AN ANALYSIS OF SOME FEATURES OF THE GRIB CODE

Harry R. Glahn

1. INTRODUCTION

The GRIdded Binary code contained in the Manual on Codes (WMO 1988) is becoming the international standard for exchanging meteorological gridpoint data and has been specified as the form of exchange of such data between the U.S. Government and the Automated Weather Interactive Processing System (AWIPS) contractor. Since a large part of the data to be transmitted in AWIPS--and especially a large part of the non-satellite data--is gridpoint, the size (number of bytes) of the (packed) message used to transmit the "information" is of considerable interest and importance. While compact in some respects, GRIB is not especially compact in other respects. This note discusses some of the features of GRIB that could be improved in future editions.[1]

Specifically, the issues addressed are (1) the use of a bit map for identifying missing data, (2) the use of full octets to pack group minima in second-order packing, (3) the use of a bit map to identify the "groups" for second-order packing, (4) the scanning modes, and (5) the limitation of packing the actual scaled gridpoint values rather than taking full advantage of the spatial redundancy in the meteorological fields by packing differences between gridpoints. These five issues are discussed in approximately reverse order of importance.

The discussion will be limited to (1) where there is only one value per gridpoint, not a vector, as specified in bit 6 of GRIB Code Table 11, and (2) gridpoint data in reference to a particular map projection such as polar stereographic (not spherical coordinates). These comments may also be relevant to vectors at gridpoints and latitude/longitude grids, but no attention has been given to them.

Certain aspects of the GRIB code have been previously studied and documented in TDL Office Note 92-11 (Glahn 1992); reference to it here will be by TDLON92-11. The 1992 office note should be used as a companion to this one.

2. BIT MAP FOR MISSING POINTS IN A RECTANGULAR GRID

There are two ways the basic grid can be specified in the GRIB code--by including the full description in Section 2 and by a Grid Definition Number (octet 7 of Section 1). In the latter case, Section 2 can be omitted and the receiver would then have to know the definition of the grid (size, location, gridpoint spacing, etc.--the information that would have been included in Section 2). While it is possible the Grid Definition Number could define a number of points in relation to a particular map projection that didn't form a rectangle, the grid description in Section 2 allows only a rectangular grid. However, there could be "missing points"--points which could be interior points; points which could, in effect, redefine the rectangle's boundary; or points usually present but with missing data on a particular occasion.

---

[1]The current edition is 1.

If there are indeed "missing" gridpoint values, GRIB carries this information through a bit map--that is, a 1 or 0 is packed for each and every gridpoint representing, respectively, whether or not a value exists at that point. If the value is zero, that point is "skipped" in all processing and must be recognized as missing by the receiver. While one bit per point seems small, it may in actuality not be small in relation to message size if only, say, five bits are required on average to send the data at points with data.

An alternate way of sending the information regarding missing values would be to reserve the maximum value (all 1 bits) within the field to represent missing. Without knowing the specific situation, it's not possible to know which would be the more efficient (require less bits) method. If only a few values were missing, as might be the case with a nearly rectangular grid but a few missing points, the "all ones" solution would be better. However, if, say, 20% of the values were characteristically missing, the bit map would probably be better.

Since most uses of GRIB will likely be for fully populated rectangular grids (and, therefore, no bit map is needed), and when there <u>are</u> missing values the number of such values may be large with respect to the rectangle, the bit map may be the better overall solution in most cases. A change to include the "all ones" option is not recommended.[2]

3. FULL OCTET FOR DEFINING GROUP WIDTHS

A current capability of GRIB is to not only reduce the range of the scaled[3] values by subtracting the overall grid minimum value[4] (which renders all

---

[2]One use for non-rectangular grids in AWIPS is the sending by a Weather Forecast Office (WFO) of the official forecasts in gridpoint form over (only) that WFO's area of responsibility to other WFO's and River Forecast Centers. These forecasts would be mosicked with other such forecasts onto a larger map, such as the AWIPS Local Area, an approximate 750 by 750 km area.

[3]Scaling by a power of 10 and/or a power of 2 is available.

[4]It is noted that this "reference" value in octets 7-10 of Section 4 is floating point, and that it is the only such number needed in the GRIB message when the scaled values are integers. Since different computer systems use different representations for floating point numbers, this seems to be an unnecessary and disagreeable complication. If the reference value is to be packed with more accuracy than the scaled gridpoint values (with the reference value subtracted), then the reference value can be dealt with more easily by packing four integer values: a positive value representing the absolute value of the scaled reference value (in, say, 23 bits), its sign (in one bit), another positive integer representing the absolute value of a power of 2 (in, say, 7 bits), and its sign (in one bit). The power of 2 and its sign would be used as an exponent to the first two of these four values. Each of these positive integer values can be unpacked separately and easily dealt with. No knowledge would be required of the bit configuration of floating point numbers of the computer on which the code is run, thereby making the code transportable. It should be easier, and reduce errors, for someone using, say, VAX equipment to deal with this arrangement than to compose the reference value into a floating point representation not matching the VAX.

resulting values positive or zero),[5] but to also further reduce the range of values within groups of "adjacent" gridpoints by subtracting each group's minimum from all members of that group.

The efficiency of that process for reducing message size depends on the spatial redundancy of the data. In many fields of data, such as 500-mb height or surface temperature, adjacent values may well be more like each other in size than like values in another group at another place on the grid. Other fields having more small scale spatial variability may not benefit much, if at all.

This option is well worth implementing, but does require a method of determining groups in such a manner to take good advantage of the option. The groups are determined by scanning the data, point-to-point as defined in the scanning mode flags in octet 28 of Section 2. An algorithm for defining groups is contained in TDLON92-11.

GRIB provides for packing the minimum of each of the, say, LX groups in an efficient manner--transmitting the number of bits needed for the largest minimum (octet 11 of Section 4), then packing each value in that number of bits. The number of bits needed can be easily determined, once the minima are known.

The current GRIB code uses a full octet for each group to define the "width" of the group (i.e., the number of bits needed to pack the maximum of the group after the group minimum is subtracted). This is wasteful in that a group will characteristically require 5 to 10 bits to contain the necessary information, which would take only 4 bits to define. That is, even if the maximum value in any group requires 15 bits, only half an octet would be needed to carry the width information.

As explained earlier in this section, GRIB uses only as many bits as necessary to send the LX group minima. This same procedure can also be used for group size. Several tests were done with 83 X 59 eta model (Mesinger et al. 1990) fields[6] in which the number of bytes required for this proposed procedure was compared to the number of bytes required by the current GRIB code. The "new" code was exactly the same as GRIB Edition 1 except that instead of the widths starting in octet 22 of Section 4 and occupying an octet each, octet 22 contained the "width of widths" (that is, the number of bits necessary to define the widths), followed by P1 (octets 17-18) widths of octet 22 size.

Table 1 gives results for four different fields--500-mb height, 700-mb omega, precipitation amount, and surface temperature. The data for these fields were rounded to meters, tenths of microbars sec^{-1}, hundredths of inches, and tenths of degrees Celsius, respectively. For an additional comparison, the 500-mb heights were rounded to tenths of meters.

---

[5]This will not reduce the range when the original values vary about zero, but the bits necessary to pack the resulting positive numbers may be less (see TDLON92-11, footnote 7).

[6]These same fields were used in TDLON92-11, and maps of them are presented there.

Table 1. The number of bytes required for the GRIB message for each of five tests and six packing options. The units for the two 500-mb tests were meters and tenths of meters, respectively. See text for other units.

Field	Grib Ed. 1	Width of Widths	No Bit Map	Scan Option	2nd Order Differences	Range Option
500-mb height	4717	4591	4170	4117	2308	2308
500-mb height	6879	6749	6334	6245	4248	4248
700-mb omega	4577	4452	4029	3980	4102	3980
Precipitation	2877	2752	2329	2183	2895	2183
Surface temp	5083	4950	4539	4333	4265	4333
Total	24133	23494	21401	20858	17818	17052

The "grouping" algorithm used is that given in TDLON92-11 with a minimum group size (MINPK) of 15 and an increment (INC) of 3 (again, see TDLON92-11 for further explanation). The number of groups generated was fairly constant and ranged from 251 to 268 for the five tests. In each test, the number of bytes required by the proposed method (given in the column headed "Width of widths") was reduced from that of GRIB Edition 1, and based on the totals, the reduction was about 3%. While not impressive, this (almost) guaranteed improvement is very easy to achieve with only a minor adjustment to the code and essentially no more processing at either the sending (packing) or receiving (unpacking) end.

4. BIT MAP FOR DEFINING 2ND ORDER GROUPS

GRIB must also contain information about the size of each group, or alternatively where each group starts. GRIB uses a bit map for this purpose. This map, contained in Section 3 of the GRIB message, has a 1 for each gridpoint where a group starts and a 0 at each other point. As with the bit map discussed in Section 2 of this document, one bit is required for each gridpoint with data. This means that if a group has 15 values in it, 15 bits will be required to send the size of group information. A better way to do this is to pack the size of each group (along with the minimum of each group) and send these sizes in as few bits as possible--the number of bits required is easily determined once the group sizes are known. If no group is over 63 in size, then each group size can be sent in 6 bits; this can be compared to the 15 bits needed in a bit map for a 15-value group.

These two methods of sending group size information has been tested on the 83 X 59 fields from the eta model discussed previously. The three differences in the code used for testing were (1) the variable width possibility described and tested in Section 3 was used (that is, this test includes the option tested in Section 3), (2) immediately after the group widths information, 8 bits were used to define the number of bits needed (say MBIT) to provide the group size information, followed by P1 (octets 17-18) group sizes, each in MBIT's, and (3) the bit map was deleted. To keep continuity with the current GRIB, binary zeros were used as needed to pad to a whole octet.

Table 1 shows in the column headed "No Bit Map" the bytes needed for this option. In each test, the bytes required was reduced from the previous test, and based on the totals, the reduction over using a bit map was about 9%. This guaranteed improvement would require more change in the GRIB code than the one proposed in Section 3, but the processing time should be a wash at both ends.

5. SCAN MODES

GRIB provides for scanning in either the plus or minus IX direction and in either the plus or minus JY direction, and with either IX or JY points in sequence (consecutive) (octet 28 of Section 2--Flag/Code Table 8). When packing by groups, it is better to reverse the order on alternate rows (when IX points are consecutive) or columns (when JY points are consecutive). With the current code, when IX points are consecutive, one marches along a row from start to finish, then jumps back to the start of the next row. The two points spanning the end of one row and the beginning of the next are not neighbors. By reversing the marching order for alternate rows, points will always be adjoining.

To implement this option, no change to the GRIB code is necessary, but the Flag/Code in octet 28 of Section 2 must be augmented and the software modified slightly. Since only three bits are currently in use in octet 28, the fourth can be used so that a 1 indicates reversal of alternate rows (or columns).

This option was tested, along with the two described in the previous two sections. That is, this test included (1) a variable width of widths, (2) the elimination of the secondary bit map in favor of sending explicitly the number in each group, and (3) this "alternate reversal" method.

Table 1 gives the results in the column headed "Scan Option." In each test, the number of bytes required was reduced, and based on the totals, the reduction was about an additional 3%. To implement this feature would require less than 10 extra lines of code for each the packer and unpacker.

6. SECOND-ORDER SPATIAL DIFFERENCES

Many meteorological fields are quite redundant, point-to-point. Although much of this redundancy can be eliminated by subtracting group means, considerable redundancy will still remain for smooth fields--those with little (compared to the overall variability) small-scale detail. A method of removing this redundancy is to not pack each value (with the appropriate minimum removed) but to pack the second-order differences between gridpoints. To recover the original, say, N values, one must have not only the N-2 second-order differences, but also the first value, and the first first-order difference. This process is explained in more detail in TDLON92-11 and is not repeated here.

This method was tested, along with the options in Sections 3, 4, and 5, and the results are in the penultimate column in Table 1. Here, improvement was not achieved for all fields. The smooth field, 500-mb height, was helped considerably--about 32% for tenths of meters resolution and 44% for meter resolution. The relatively smooth field of surface temperature benefitted slightly--about 2%. The other two fields contained much small scale detail

and did not benefit.[7] In fact, the precipitation field required more bytes than the GRIB code, Edition 1.

It is likely that a majority of meteorological fields that one would be packing would be smooth, especially forecasts at various levels of the atmosphere produced by numerical models. The question is, can second-order packing be used where it is best and not where is isn't. The answer is a qualified "yes." One could, of course, go through the whole packing process twice, once with second-order differences and once without, then use the method that required less bytes. But this would require approximately double the packing (cpu) time of either method by itself. An alternative is to use a "cheap" algorithm to estimate which is the better procedure.

One such algorithm is to compute the average range of numbers over groups of size MINPK for both the original values and the second-order differences. The lower average range would indicate the better procedure. This was tried, and the results are given in the last column of Table 1. The algorithm gave the correct solution in all cases except for the surface temperature. Here, it indicated to not use second-order differencing when, in fact, that would have been better. But the improvement in that case would have been quite small (less than 2%). It seems the algorithm gives the correct solution when the difference is major (important) and may or may not give the correct answer when the difference is small (and it doesn't much matter which method is used).

Another way to implement this option is to predetermine, through a few simple tests, which types of fields are not likely to benefit from second-order spatial packing.

7. SUMMARY AND CONCLUSIONS

The overall improvement on these five test fields, computed on the totals of the first and last digital columns, was 29%. The range was from 51% for 500-mb height rounded to meters to 13% for 700-mb omega. The overall improvement for a full range of synoptic-scale analyses and forecasts would likely be about 35% to 40%.

The current version of Appendix K to the AWIPS Systems Requirements Specification[8] estimates the total point to multipoint gridpoint traffic to be 713 megabytes and the peak traffic (over four 50-minute periods) due to gridpoint data from the mesoscale model to be 248 kilobits per second. If a 35% overall reduction could be achieved, these values would reduce to 463 megabytes and 161 kilobits per second, respectively.

---

[7]As stated in TDLON92-11, these eta model fields were chosen for experimentation because they had fine-scale detail to see how possible packing options would be affected by detail or lack thereof. Note that these results agree comparatively with those in TDLON92-11, Table 6, columns headed "Method 5" and "Method 8." The values are not the same because a full GRIB code was not used in TDLON92-11.

[8]Appendix K is currently undergoing revision and the estimated transmission totals and rates will change. However, these comparisons should be representative of the magnitudes involved.

The changes to implement the options described in Sections 3 through 6 above would not change the overall structure of Edition 1 of the GRIB code. As mentioned previously, the packing of the group widths and sizes would follow exactly the same pattern now used in GRIB for packing the group minima, whereas the current GRIB packs each of these kinds of information differently--variable number of bits for group minima, constant number of bits (8) for group widths, and a bit map for group membership.

The alternate scanning can be implemented by only the extension to bit 4 of octet 28 in GRIB Section 2. The use of second-order spatial differences requires more change. For a field with N points, N values are still packed--the first value in the field, the first first-order difference, and N-2 second-order differences.

In order to get an indication of the added computer time required for the options recommended here, the 500-mb height field was packed (and then unpacked) to tenths of meters (row 2, Table 1) 50 times for both GRIB Edition 1 (column 1, Table 1) and the recommended options (last column, Table 1). This included reading the height field and writing the packed record, as well as reading some control information that is needed for each individual field and writing a half dozen lines of diagnostic print for each field. The clock time on a VAX 3400--approximately a 1 MIP (millions of instructions per second) computer--was 1.8 seconds per 83 X 59 field for GRIB Edition 1 and 1.4 seconds per field for the options recommended here. The unpacking--which included reading the packed field but not writing the unpacked field, and writing a half dozen lines of diagnostic information--took 0.8 and 0.6 seconds per field for GRIB Edition 1 and the recommended options, respectively. These values indicate that the added cpu time is not only dwarfed by the input/output (I/O) normally required in such processes, but that the smaller packed record (output for packing, input for unpacking) for the recommended options (4,248 vs 6,879 bytes, a 38% reduction) actually made the recommended process speedier in terms of clock time. The I/O used for the data records was normal FORTRAN binary reads and writes; it is recognized that faster I/O methods may be practical. The point here is that the difference in cpu time seems unimportant.

Appendix 1 (2) gives a FORTRAN packing (unpacking) code for implementing these options. These codes essentially implement grids with relation to polar stereographic and Lambert map projections. Not all GRIB options are implemented by these routines. Appendix III summarizes the changes that would be necessary in the Manual on Codes (WMO 1988) to implement the options recommended in this office note and contained in the code in Appendices I and II.

REFERENCES

Glahn, H. R., 1992: On the packing of gridpoint data for efficient transmission. TDL Office Note 92-11, National Weather Service, NOAA, U.S. Department of Commerce, 32 pp.

Mesinger, R., T. L. Black, D. W. Plummer, and J. H. Ward, 1990: Eta model precipitation forecasts for a period including tropical storm Allison. WEA Forecasting, 5, 483-493.

WMO, 1988: Manual on Codes, Vol. 1, Part B--Binary Codes. WMO No. 306, World Meteorological Organization, Geneva.

APPENDIX I

This appendix contains a listing of the FORTRAN code to implement the options recommended in this office note. It consists of four routines, GRIBC, BGVRBL, GRIBPR, and PKBG, the latter three being called by the former. Another routine called by GRIBC, Q9VI32, is for use only when implementing this code on a VAX computer, and is not included. Subroutine BGVRBL corresponds closely to VRBLPK in TDLON92-11.

```
      SUBROUTINE GRIBC(KFIL10,KFIL12,A,IA,IC,NX,NY,IS1,IS2,IS4,ND8,
     1                 LBIT,JMAX,JMIN,NOV,IPACK,ND4,MINPK,INC,
     2                 LX,MAXMAX,IOCTET)
C
C        APRIL 1993   GLAHN   TDL   MICROVAX
C
C        PURPOSE
C            SUBROUTINE TO PACK DATA AT "UNITS" RESOLUTION
C            PROVIDED IN A( , ) AFTER MULTIPLYING BY 10**IS1(27).
C            THE SMALLEST VALUE IS SUBTRACTED TO MAKE ALL VALUES POSITIVE.
C            ADDITIONAL VALUES ARE TAKEN OUT AT NONUNIFORM STEPS WITH
C            A MINIMUM GROUP SIZE OF MINPK.  THIS VERSION IS FOR TESTING:
C            (1) LESS THAN FULL OCTET FOR PACKING GROUP WIDTHS,
C            (2) THE PACKING OF GROUP SIZES RATHER THAN A BIT MAP,
C            (3) THE REVERSAL OF ALTERNATE ROWS, AND
C            (4) THE OPTION OF SECOND-ORDER SPATIAL DIFFERENCES.
C            THESE OPTIONS ARE EXPLAINED IN MORE DETAIL IN TDL OFFICE
C            NOTE 93-XX.  GRIB SECTIONS 0, 1, 2, 4, AND 5 ARE PACKED
C            AS EXPLAINED IN THE WMO GRIB CODE, MODIFIED AS
C            INDICATED IN THIS OFFICE NOTE.
C
C        RESTRICTIONS:  THIS SUBROUTINE DEALS WITH
C            - ONLY RECTANGULAR GRIDS WITH RESPECT TO NORTH POLAR
C              STEREOGRAPHIC OR LAMBERT CONFORMAL MAP PROJECTIONS
C              (CODE TABLE 11, BIT 1 = 0; CODE TABLE 6 = 3 OR 5.
C            - ONLY SINGLE VALUES AT GRIDPOINTS, NOT MATRICES; CODE
C              TABLE 11, BIT 6 = 0.
C            - SCALES BY POWERS OF 10, NOT POWERS OF 2.
C            - SECTION 1 IS ALWAYS 28 OCTETS LONG.
C            - SECTION 2 IS 28 OCTETS LONG FOR POLAR STEREOGRAPHIC
C              AND 42 OCTETS LONG FOR LAMBERT.
C
C        ERROR RETURNS
C              IF SUBROUTINE PKBG DETECTS AN ERROR, RETURN IS MADE TO
C              FORTRAN STATEMENT 900 WHERE A DIAGNOSTIC IS PRINTED AND
C              RETURN IS TO THE CALLING PROGRAM WITH THE ERROR NUMBER IN
C              IER.
C
C        DATA SET USE
C           KFIL10 - UNIT NUMBER FOR CURRENT CONSOLE. (OUTPUT)
C           KFIL12 - UNIT NUMBER FOR OUTPUT (PRINT) FILE. (OUTPUT)
C
C        VARIABLES
C             KFIL10 = UNIT NUMBER FOR CURRENT CONSOLE.  (INPUT)
C             KFIL12 = UNIT NUMBER FOR OUTPUT (PRINT) FILE.  (INPUT)
C           A(IX,JY) = ARRAY FOR ORIGINAL INPUT GRID (IX=1,NX) (JY=1,NY).
```

```
C                    (INPUT)
C          IA(IX,JY) = ARRAY FOR SCALED AND ROUNDED INPUT GRID
C                    (IX=1,NX) (JY=1,NY).  (INTERNAL)
C              IC(K) = WORK ARRAY (K=NX*NY).  (INTERNAL)
C              NX,NY = DIMENSIONS OF A( , ), IA( , ), AND
C                    IC( ).  (INPUT)
C             IS1(L) = HOLDS THE OCTETS TO FURNISH FOR GRIB
C                    SECTION 1 (L=1,ND8).  (INPUT)
C                    THE FOLLOWING LOCATIONS CORRESPOND TO OCTET
C                    NUMBERS AND ARE INPUT:
C                    IS1(4) = GRIB VERSION NUMBER OF CODE TABLE 2.
C                    IS1(5) = CODE TABLE 0, ORIGINATING OFFICE.
C                    IS1(6) = GENERATING PROCESS NUMBER.
C                    IS1(7) = CATALOGUE NUMBER OF GRID FOR CENTER
C                             DEFINED IN IS1(5).  (EVEN IF USED,
C                             SECTION 2 WILL ALWAYS BE PRESENT.)
C                    IS1(8) = CODE TABLE 1 = 10000000 BINARY;
C                             SECTION 2 ALWAYS PRESENT, SECTION 3
C                             NOT PRESENT.
C                    IS1(9) = CODE TABLE 2, PARAMETER DEFINITION
C                    IS1(10) = CODE TABLE 3 VALUE
C                    IS1(11) = CODE TABLE 3 VALUE
C                    IS1(13) = YEAR OF CENTURY
C                    IS1(14) = MONTH
C                    IS1(15) = DAY
C                    IS1(16) = HOUR
C                    IS1(17) = MINUTE
C                    IS1(18) = CODE TABLE 4, UNIT OF TIME RANGE
C                    IS1(19) = PROJECTION TO END OF PERIOD
C                    IS1(20) = TIME INTERVAL
C                    IS1(21) = CODE TABLE 5, TIME RANGE INDICATOR
C                    IS1(22) = NUMBER INCLUDED IN AVERAGE, IF
C                              RELEVANT
C                    IS1(24) = NUMBER MISSING FROM AVERAGE, IF
C                              RELEVANT
C                    IS1(25) = CENTURY OF REFERENCE.
C                    IS1(27) = DECIMAL SCALE FACTOR.
C             IS2(L) = HOLDS THE OCTETS FOR GRIB SECTION 2 (L=1,ND8).
C                    NOT ALL LOCATIONS ARE USED.  (INPUT)
C                    THE FOLLOWING LOCATIONS CORRESPOND TO OCTET
C                    NUMBERS AND ARE INPUT:
C                    IS2(5) = 255 (SINGLE VALUES AT GRIDPOINTS)
C                    IS2(6) = CODE TABLE; MAP PROJECTION = 3 OR 5
C                             ONLY
C                    IS2(7) = NUMBER OF POINTS IN IX DIRECTION
C                    IS2(9) = NUMBER OF POINTS IN JY DIRECTION
C                    IS2(11) = LATITUDE OF FIRST GRIDPOINT,
C                              MUST BE POSITIVE
C                    IS2(14) = LONGITUDE OF FIRST GRIDPOINT,
C                              MUST BE POSITIVE
C                    IS2(17) = CODE TABLE 7
C                    IS2(18) = ORIENTATION OF GRID, MUST BE POSITIVE
C                    IS2(21) = DX = IX DIRECTION GRID LENGTH
C                    IS2(24) = DY = JY DIRECTION GRID LENGTH
C                    IS2(28) = CODE TABLE 8, SCANNING MODE = 4 OR 5
```

```
C                              ONLY
C                     IS2(29) = LATITUDE OF FIRST SECANT CUT FOR
C                               LAMBERT MAP ONLY
C                     IS2(32) = LATITUDE OF SECOND SECANT CUT FOR
C                               LAMBERT MAP ONLY
C                     IS2(35) = LATITUDE OF SOUTHERN POLE FOR LAMBERT
C                               MAP ONLY
C                     IS2(38) = LONGITUDE OF SOUTHERN POLE FOR LAMBERT
C                               MAP ONLY
C               IS4(L) = HOLDS THE OCTETS FOR GRIB SECTION 4 (L=1,ND8).
C                        ONLY 2 LOCATIONS ARE USED.
C                        THE FOLLOWING LOCATIONS CORRESPOND TO OCTET
C                        NUMBERS:
C                        IS4(4) = CODE TABLE 11; FLAG = 01110000 BINARY
C                                 INDICATING GRIDPOINT DATA, SECOND-ORDER
C                                 PACKING, INTEGER VALUES, AND ADDITIONAL
C                                 FLAGS IN OCTET 14.  (INTERNAL)
C                        IS4(14) = CODE TABLE 11; FLAG = 0001X000 BINARY
C                                  INDICATING SINGLE DATUM PER GRIDPOINT,
C                                  NO SECONDARY BIT MAP, AND SECOND-ORDER
C                                  VALUES HAVE DIFFERENT WIDTHS.  A 0 OR
C                                  1 IN THE FIFTH FROM THE LEFT (WHERE THE
C                                  X IS) INDICATES THAT SECOND-ORDER SPATIAL
C                                  DIFFERENCES ARE OR ARE NOT PACKED,
C                                  RESPECTIVELY.  (INTERNAL)
C                   ND8 = DIMENSION OF IS1( ), IS2( ), AND IS4( ).  (INPUT)
C                LBIT(M) = THE NUMBER OF BITS NECESSARY TO HOLD THE
C                          PACKED VALUES FOR EACH GROUP M (M=1,LX).
C                          (OUTPUT)
C                JMAX(M) = THE MAXIMUM OF EACH GROUP M OF PACKED VALUES
C                          AFTER SUBTRACTING THE GROUP MINIMUM VALUE
C                          (M=1,LX).  (OUTPUT)
C                JMIN(M) = THE MINIMUM VALUE SUBTRACTED FOR EACH GROUP
C                          M (M=1,LX).  (OUTPUT)
C                 NOV(M) = THE NUMBER OF VALUES IN GROUP M (M=1,LX).
C                          (OUTPUT)
C               IPACK(J) = THE ARRAY TO HOLD THE ACTUAL PACKED MESSAGE
C                          (J=1,MAX OF ND4).  (OUTPUT)
C                   ND4 = THE SIZE OF THE ARRAYS JMAX( ), LBIT( ),
C                         NOV( ), JMIN( ), AND IPACK( ).  (INPUT)
C                 MINPK = VALUES ARE PACKED IN GROUPS OF MINIMUM SIZE
C                         MINPK.  ONLY WHEN THE NUMBER OF BITS NEEDED TO HANDLE
C                         A GROUP CHANGES WILL A NEW GROUP BE FORMED.  (INPUT)
C                   INC = NUMBER OF VALUES TO ADD TO THE GROUP TO BE
C                         PACKED AT A TIME.  (INPUT)
C                    LX = THE NUMBER OF GROUPS (THE NUMBER OF 2ND ORDER
C                         MINIMA).  (OUTPUT)
C                 MAXMAX = THE MAXIMUM VALUE IN IC( ), WHICH IS THE
C                          LARGEST VALUE THAT HAS TO BE PACKED.  (OUTPUT)
C                 IOCTET = THE TOTAL MESSAGE SIZE IN OCTETS.  (OUTPUT)
C                  SCALE = SCALING PARAMETER = 10.**IS1(27).  (INTERNAL)
C              MAXA,MINA = THE MAXIMUM AND MINIMUM VALUES IN IA( , ) AND
C                          IC( ) BEFORE SUBTRACTING THE MINIMUM VALUE.
C                          (INTERNAL)
C                    LOC = WORD POSITION IN IPACK( ) WHERE THE FIRST
```

```
C                (LEFTMOST) BIT OF THE VALUE TO PACK GOES.
C                PKBG UPDATES IT.
C          IPOS = BIT POSITION WITHIN IPACK(LOC) WHERE THE FIRST
C                (LEFTMOST) BIT OF THE VALUE TO PACK GOES.
C                PKBG UPDATES IT.
C          IBIT = THE NUMBER OF BITS REQUIRED TO PACK THE GROUP
C                MINIMUM VALUE.  (INTERNAL)
C          KBIT = THE NUMBER OF BITS REQUIRED TO PACK THE WIDTH
C                OF WIDTHS--OCTET 22, SECTION 4.  (INTERNAL)
C          MBIT = THE NUMBER OF BITS REQUIRED TO PACK THE
C                GROUP SIZES.  (INTERNAL)
C          IBIT = THE NUMBER OF BITS REQUIRED TO PACK THE
C                FIRST VALUE IN THE FIELD (WITH THE REFERENCE
C                REMOVED).  USED ONLY WHEN PACKING SECOND ORDER
C                DIFFERENCES.
C          JBIT = THE NUMBER OF BITS REQUIRED TO PACK THE
C                ABSOLUTE VALUE OF THE FIRST FIRST ORDER
C                DIFFERENCE.  USED ONLY WHEN PACKING SECOND ORDER
C                DIFFERENCES.
C          NBIT = THE NUMBER OF BITS REQUIRED TO PACK THE
C                ABSOLUTE VALUE OF THE MINIMUM SECOND ORDER
C                DIFFERENCE.  USED ONLY WHEN PACKING SECOND ORDER
C                DIFFERENCES.
C        IFIRST = THE FIRST VALUE IN THE FIELD TO PACK (WITH THE
C                REFERENCE REMOVED).  USED ONLY WHEN PACKING SECOND
C                ORDER DIFFERENCES.
C         IFOD = THE FIRST FIRST ORDER DIFFERENCE.  USED ONLY WHEN
C                PACKING SECOND ORDER DIFFERENCES.
C       FMNVAX = VAX FLOATING POINT REPRESENTATION OF MINA,
C                THE REFERENCE VALUE.
C       FMNIBM = IBM FLOATING POINT REPRESENTATION OF MINA,
C                THE REFERENCE VALUE.  THIS IS THE REPRESENTATION
C                SPECIFIED FOR GRIB.  CONVERSION FROM FMNVAX TO
C                FMNIBM DONE BY SUBROUTINE Q9VI32.
C          NXY = NX*NY.  MAY BE SET TO NXY-2 BY GRIBPR.  (INTERNAL)
C       NCOUNT = COUNTS THE VALUES ACTUALLY PACKED.  THIS CAN BE
C                LESS THAN NXY WHEN ONE OR MORE GROUPS HAVE THE
C                SAME VALUE, AND THE VALUES ARE OMITTED.
C        C7777 = HOLDS '7777'.  EQUIVALENCED TO I7777 FOR
C                PROVISION TO SUBROUTINE PKBG AS AN INTEGER.
C                (INTERNAL)  (CHARACTER*4)
C        I7777 = SEE C7777.  (INTERNAL)
C        GRIBX = HOLDS 'GRIB'.  EQUIVALENCED TO IGRIB FOR
C                PROVISION TO SUBROUTINE PKBG AS AN INTEGER.
C                (INTERNAL)
C        IFILL = NUMBER OF BITS TO PAD MESSAGE (AT THAT POINT
C                IN THE PROCESS) TO AN EVEN OCTET.  (INTERNAL)
C       MOCTET = NUMBER OF OCTETS CALCULATED FOR LENGTH OF
C                SECTIONS, ETC.  (INTERNAL)
C        STATE = HOLDS 4 CHARACTERS FOR PRINTOUT IN CASE OF
C                ERROR.  (INTERNAL)  (CHARACTER*4)
C
C     NON SYSTEM SUBROUTINES CALLED
C        BGVRBL, PKBG, GRIBPR, Q9VI32,
C
```

```
      CHARACTER*4 GRIBX/'GRIB'/,
     1            C7777/'7777'/
      CHARACTER*4 STATE
      LOGICAL SECOND
C
      DIMENSION A(NX,NY),IA(NX,NY),
     1          IC(NX*NY)
      DIMENSION JMAX(ND4),JMIN(ND4),NOV(ND4),LBIT(ND4),IPACK(ND4)
      DIMENSION IS1(ND8),IS2(ND8),IS4(ND8)
      EQUIVALENCE (IGRIB,GRIBX),(C7777,I7777)
C
      DATA IZERO/0/,
     1     IONE/1/

      NXY=NX*NY
C
C        IPACK( ) MUST BE ZEROED.
C
      DO 90 K=1,ND4
      IPACK(K)=0
 90   CONTINUE
C
C        DATA ARE IN A( , ).  NOW SCALE THEM INTO IA( , ).
C
      SCALE=10.**IS1(27)
C
      DO 103 JY=1,NY
      DO 102 IX=1,NX
      IA(IX,JY)=NINT(A(IX,JY)*SCALE)
 102  CONTINUE
 103  CONTINUE
C
      IF(IS2(28).EQ.5)GO TO 120
      IF(IS2(28).EQ.4)GO TO 110
C        THE ONLY SCANNING MODES ALLOWED ARE REPRESENTED BY IS2(28)
C        EQUAL 4 OR 5.  EACH DESIGNATES LEFT TO RIGHT FIRST, THEN
C        BOTTOM TO TOP.  ADDITIONALLY, 5 MEANS TO REVERSE THE ORDER
C        ON ALTERNATE ROWS TO TAKE ADVANTAGE OF THE 2ND ORDER PACKING.
      WRITE(KFIL12,105)IS2(28)
 105  FORMAT('0SCANNING MODE NOT PROVIDED FOR.  IS2(28) ='I6,'. ',
     1       ' STOP IN GRIBC AT 105.')
      STOP 105
C
C        PUT DATA INTO A SINGLE DIMENSIONED ARRAY.
C
 110  K=0
C
      DO 115 JY=1,NY
      DO 114 IX=1,NX
      K=K+1
      IC(K)=IA(IX,JY)
 114  CONTINUE
 115  CONTINUE
C
      GO TO 127
```

```
C
C        PUT DATA INTO A SINGLE DIMENSIONED ARRAY.  THE FIRST ROW IS
C        PROCESSED LEFT TO RIGHT, THEN HOP UP TO ROW 2 AND PROCEED
C        RIGHT TO LEFT, ETC.  THIS MAKES FOR SMALLER DIFFERENCES.
C
 120  K=0
C
      DO 125 JY=1,NY
      DO 124 IX=1,NX
      K=K+1
      I=IX
      IF(MOD(JY,2).EQ.0)I=NX+1-IX
      IC(K)=IA(I,JY)
 124  CONTINUE
 125  CONTINUE
C
C        FIND THE MAX AND MIN VALUES, MAXA AND MINA.
C
 127  MAXA=IC(1)
      MINA=IC(1)
C
      DO 130 K=2,NXY
      IF(IC(K).GT.MAXA)MAXA=IC(K)
      IF(IC(K).LT.MINA)MINA=IC(K)
 130  CONTINUE
C
C        ADJUST VAUES TO MAKE THEM POSITIVE IN IC( ) AND TO MAKE THE SMALLEST
C        EQUAL 0.  ALSO, FIND THE MAMIMUM, MAXMAX, AFTER ADJUSTMENT.
C
      DO 140 K=1,NXY
      IC(K)=IC(K)-MINA
 140  CONTINUE
C
      MAXMAX=MAXA-MINA
C
C        CALL GRIBPR TO COMPUTE SECOND ORDER DIFFERENCES AND DECIDE
C        WHETHER TO USE THEM IN PACKING.
C
      IDIM=NXY
      SECOND=.FALSE.
      IF(MINPK.GT.NXY-2)GO TO 145
      CALL GRIBPR(KFIL10,KFIL12,IC,IA,A,IDIM,NXY,MINPK,
     1            IMIN,IFIRST,IFOD,SECOND)
C        UPON RETURN, SECOND IS TRUE WHEN SECOND ORDER DIFFERENCES
C        ARE TO BE PACKED.  NXY HAS BEEN SET TO NXY-2, AND THESE NXY
C        DIFFERENCES MINUS THEIR MINIMUM VALUE IMIN ARE NOW IN IC( )
C        IF SECOND IS FALSE, NXY AND IC( ) HAVE NOT BEEN MODIFIED.
C
C        CALL VRBLPK TO CALCULATE LX, JMIN( ), JMAX( ), LBIT( ),
C        AND NOV( ).
C
 145  CALL BGVRBL(KFIL10,KFIL12,IC,NXY,MINPK,INC,
     1            JMIN,JMAX,IBIT,LBIT,NOV,ND4,LX)
C
```

```
C         SUBTRACT LOCAL MIN FOR EACH OF LX GROUPS.
C
      K=0
C
      DO 153 L=1,LX
      DO 152 M=1,NOV(L)
      K=K+1
      IC(K)=IC(K)-JMIN(L)
  152 CONTINUE
  153 CONTINUE

C         PACK SECTION 0 OF THE MESSAGE INTO IPACK( ).
C
      STATE='0   '
      LOC=1
C         LOC = 4-BYTE WORD POSITION IN IPACK( ) TO START PACKING.
C         PKBG UPDATES IT.
      IPOS=1
C         IPOS = BIT POSITION IN IPACK(LOC) TO START PUTTING VALUE.
C         PKBG UPDATES IT.
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IGRIB,32,IER,*900)
         LOC0=LOC
         IPOS0=IPOS
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,24,IER,*900)
C         OCTETS 5-7 MUST BE FILLED IN LATER; ABOVE STATEMENT HOLDS
C         THE PLACE.  LOC0 AND IPOS0 HOLD THE LOCATION.
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IONE,8,IER,*900)
C         THIS IS GRIB EDITION 1.
C
C         PACK SECTION 1 OF THE MESSAGE INTO IPACK( ).
C
      STATE='1   '
      IS1(1)=28
C         LENGTH OF SECTION 1 IS 28 OCTETS.
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS1(1),24,IER,*900)
C         LENGTH OF SECTION 1 IS 28 OCTETS.
C
      DO 210 K=4,10
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS1(K),8,IER,*900)
  210 CONTINUE
C
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS1(11),16,IER,*900)
C
      DO 220 K=13,21
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS1(K),8,IER,*900)
  220 CONTINUE
C
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS1(22),16,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS1(24),8,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS1(25),8,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,8,IER,*900)
C         A NEGATIVE DECIMAL SCALE FACTOR MUST HAVE THE MINUS SIGN,
C         IF PRESENT, IN BIT 1 OF THE 16-BIT FIELD.
      IF(IS1(27).LT.0)
     1    CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IONE,1,IER,*900)
```

```
      IF(IS1(27).GE.0)
     1     CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,1,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,ABS(IS1(27)),15,
     1           IER,*900)
C
C        PACK SECTION 2 OF THE MESSAGE INTO IPACK( ).
C
      STATE='2   '
      IF(IS2(6).EQ.5)IS2(1)=32
C        LENGTH OF SECTION 2 IS 32 OCTETS FOR POLAR STEREOGRAPHIC.
      IF(IS2(6).EQ.3)IS2(1)=42
C        LENGTH OF SECTION 2 IS 42 OCTETS FOR LAMBERT.
      IF(IS1(8).EQ.128)GO TO 225
C        IS1(8) MUST BE 128 DECIMAL =  10000000 BINARY TO INDICATE
C        SECTION 2 IS INCLUDED AND SECTION 3 IS NOT.
      WRITE(KFIL12,224)
 224  FORMAT('0IS1(8) MUST BE 128.  STOP IN GRIBC AT 224.')
      STOP 224
C
 225  CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(1),24,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,8,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(5),8,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(6),8,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(7),16,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(9),16,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(11),24,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(14),24,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(17),8,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(18),24,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(21),24,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(24),24,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,8,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(28),8,IER,*900)
      IF(IS2(6).NE.5)GO TO 230
C        THIS IS POLAR STEREOGRAPHIC.
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,32,IER,*900)
      GO TO 300
C
 230  IF(IS2(6).EQ.3)GO TO 232
      WRITE(KFIL12,231)
 231  FORMAT('0IS2(6) MUST BE EITHER 3 OR 5.  STOP IN GRIBC AT 231.')
      STOP 231
C
C        THIS IS LAMBERT.  IS2(6) IS EITHER 3 OR 5.
 232  CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(29),24,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(32),24,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(35),24,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS2(38),24,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,16,IER,*900)
C
C        OMIT SECTION 3 OF THE MESSAGE.
C
 300  IS3=0
C        LENGTH OF SECTION 3 = 0 OCTETS.
C
```

```
C         PACK SECTION 4 OF THE MESSAGE INTO IPACK( ).
C
      STATE='4.0 '
      IS4(4)=112
C         112 DECIMAL = 160 OCTAL = 1110000 BINARY.
      LOC1=LOC
      IPOS1=IPOS
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,24,IER,*900)
C         OCTETS 1-3 MUST BE FILLED IN LATER; ABOVE STATEMENT HOLDS
C         THE PLACE.  LOC1 AND IPOS1 SAVE THE LOCATION.
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS4(4),4,IER,*900)
C         THE LEFTMOST 4 BITS OF OCTET 4 SHOULD BE 0111 BINARY TO
C         INDICATE GRIDPOINT DATA, COMPLEX PACKING, INTEGER VALUES,
C         AND ADDITIONAL FLAGS IN OCTET 14.  LOC2 AND IPOS2 SAVE THE
C         LOCATION FOR THE LAST 4 BITS OF OCTET 4 FOR FILLING LATER.
      LOC2=LOC
      IPOS2=IPOS
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,4,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,16,IER,*900)
      FMNVAX=MINA
C         THE REFERENCE VALUE IS PACKED AS FLOATING POINT.
C         FMNVAX IS THE VAX REPRESENTATION.
      FMNIBM=FMNVAX
C*****************
C         NOTE:  THE FLOATING POINT NUMBER TO BE PACKED IS THE
C         GRIB IBM REPRESENTATION.  IF THIS PACKING PROGRAM IS RUN
C         ON A VAX MACHINE, THEN INSERT THE FOLLOWING STATEMENT
C         TO CONVERT FROM VAX TO IBM REPRESENTATION.
      CALL Q9VI32(FMNVAX,FMNIBM,1,ISTAT)
C         FMNIBM IS THE IBM REPRESENTATION SPECIFIED FOR GRIB.
C*****************
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,FMNIBM,32,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IBIT,8,IER,*900)
C         IBIT INTO OCTET 11 IS THE NUMBER OF BITS NEEDED FOR
C         SECOND-ORDER MINIMUM VALUES.
      LOC3=LOC
      IPOS3=IPOS
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,16,IER,*900)
C         OCTETS 12-13 HAVE TO BE PROVIDED LATER.  LOC3 AND IPOS3 SAVE
C         THE LOCATION.  OCTETS 12-13 HOLD STARTING POSITION WITHIN
C         SECTION 4 OF GROUP MINIMA.
      IS4(14)=16
      IF(SECOND)IS4(14)=IS4(14)+8
C         OCTET 14 MUST BE 00100000 BINARY = 20 OCTAL = 16 DECIMAL TO
C         INDICATE A SINGLE VALUE AT EACH POINT, NO SECONDARY BIT MAP
C         PRESENT, AND SECOND ORDER VALUES ARE OF DIFFERENT WIDTHS.
C         THE ABOVE STATEMENT ADDS A BIT IN THE 5TH PLACE TO INDICATE
C         SECOND ORDER PACKING.
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IS4(14),8,IER,*900)
      LOC4=LOC
      IPOS4=IPOS
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,16,IER,*900)
C         OCTETS 15-16 HAVE TO BE PROVIDED LATER.  LOC4 AND IPOS4 SAVE
C         THE LOCATION.  OCTETS 15-16 HOLD STARTING POSITION WITHIN
C         SECTION 4 OF PACKED VALUES.
```

```
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,LX,16,IER,*900)
C        OCTETS 17 AND 18 HOLD THE NUMBER OF SECOND ORDER MINIMA.
      LOC5=LOC
      IPOS5=IPOS
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,16,IER,*900)
C        OCTETS 19 AND 20 HOLD THE NUMBER OF PACKED VALUES.  THIS IS
C        THE NUMBER OF GRID POINTS, UNLESS ONE OR MORE GROUPS IS EMPTY
C        (ALL VALUES THE SAME).  THIS MUST BE FILLED LATER, AND
C        LOC5 AND IPOS5 SAVE THE LOCATION.
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,255,8,IER,*900)
C        OCTET 21 IS RESERVED.
C
C        PACK LX WIDTHS OF GROUP MINIMA.  FIRST DETERMINE THE NUMBER OF
C        BITS NEEDED, KBIT, AND STORE IN OCTET 22.
C
      KBIT=1
C
 314  DO 315 K=1,LX
      IF(LBIT(K).LT.2**KBIT)GO TO 315
      KBIT=KBIT+1
      GO TO 314
 315  CONTINUE
C
      WRITE(KFIL12,321)KBIT
 321  FORMAT('0NUMBER OF BITS NEEDED FOR GROUP WIDTHS ='I3)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,KBIT,8,IER,*900)
C
      STATE='4.1 '
C
      DO 325 K=1,LX
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,LBIT(K),KBIT,IER,*900)
 325  CONTINUE
C
C        PACK LX GROUP SIZES.  FIRST DETERMINE THE NUMBER OF BITS
C        NEEDED, MBIT, AND STORE IN 8 BITS.
C
      MBIT=1
C
 326  DO 327 K=1,LX
      IF(NOV(K).LT.2**MBIT)GO TO 327
      MBIT=MBIT+1
      GO TO 326
 327  CONTINUE
C
      WRITE(KFIL12,329)MBIT
 329  FORMAT('0NUMBER OF BITS NEEDED FOR GROUP SIZES ='I3)
      STATE='4.2 '
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,MBIT,8,IER,*900)
C
      DO 330 L=1,LX
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,NOV(L),MBIT,IER,*900)
 330  CONTINUE
C
```

```
C        PAD WITH ZEROS TO AN EVEN OCTET.
C
      IFILL=MOD(33-IPOS,8)
      IF(IFILL.EQ.0)GO TO 332
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,IFILL,IER,*900)
C
C        PACK LX GROUP MINIMA VALUES.
C
 332  MOCTET=LOC*4-(33-IPOS)/8+1-(8+IS1(1)+IS2(1)+IS3)
C        MOCTET = NUMBER OF THE OCTET WHERE THE GROUP MIMINA START
C        WITHIN SECTION 4.
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC3,IPOS3,MOCTET,16,
     1            IER,*900)
      DO 340 L=1,LX
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,JMIN(L),IBIT,
     1            IER,*900)
 340  CONTINUE
C
      IFILL=MOD(33-IPOS,8)
      IF(IFILL.EQ.0)GO TO 342
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,IFILL,IER,*900)
C        PAD WITH ZEROS TO AN EVEN OCTET.  CAN NOW FILL OCTETS 15-16.
C        THE UPDATED POSITION IPOS IS WHERE THE SECOND ORDER DATA BEGIN.
 342  MOCTET=LOC*4-(33-IPOS)/8+1-(8+IS1(1)+IS2(1)+IS3)
C        MOCTET = NUMBER OF THE OCTET WHERE THE SECOND ORDER DATA BEGIN
C        WITHIN SECTION 4.
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC4,IPOS4,MOCTET,16,
     1            IER,*900)
C
C        PACK THE VALUES, WITH BOTH FIRST AND SECOND ORDER MINIMA OUT.
C
      STATE='4.3 '
      IF(.NOT.SECOND)GO TO 3480
C        SECOND ORDER PACKING BEING DONE, SO ADDITIONAL VALUES MUST BE
C        INSERTED BEFORE THE DATA VALUES THEMSELVES.  ONCE THESE VALUES
C        ARE INSERTED, THE VALUES IN IC( ) CAN BE PACKED JUST AS WHEN
C        SECOND ORDER VALUES ARE NOT USED.  THE NUMBER OF DATA VALUES
C        ARE, HOWEVER, REDUCED BY TWO WHEN SECOND ORDER PACKING IS DONE.
C
C        FIND THE NUMBER OF BITS (IBIT) NECESSARY TO PACK THE FIRST
C        VALUE AND PACK IBIT INTO 5 BITS AND THE FIRST VALUE.  THE FIRST
C        VALUE WILL BE POSITIVE SINCE THE REFERENCE VALUE IS STILL USED.
C
      IBIT=1
 343  IF(IFIRST.LT.2**IBIT)GO TO 344
      IBIT=IBIT+1
      GO TO 343
C
 344  CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IBIT,5,IER,*900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IFIRST,IBIT,IER,*900)
C
C        FIND THE NUMBER OF BITS (JBIT) NEEDED TO PACK THE ABSOLUTE
C        VALUE OF THE FIRST FIRST ORDER DIFFERENCE AND PACK JBIT INTO
C        5 BITS, THE SIGN INTO 1 BIT, AND THE ABSOLUTE VALUE OF THE
C        FIRST FIRST ORDER DIFFERENCE INTO JBIT BITS.
```

```
C
      JBIT=1
 345  IF(ABS(IFOD).LT.2**JBIT)GO TO 346
      JBIT=JBIT+1
      GO TO 345
C
 346  CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,JBIT,5,IER,*900)
      IF(IFOD.LT.0)CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,1,1,IER,
     1                       *900)
      IF(IFOD.GE.0)CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,0,1,IER,
     1                       *900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,ABS(IFOD),JBIT,IER,
     1          *900)
C
C        FIND THE NUMBER OF BITS (NBIT) NEEDED TO PACK THE ABSOLUTE
C        VALUE OF THE MINIMUM SECOND ORDER DIFFERENCE AND PACK NBIT
C        INTO 5 BITS, THE SIGN INTO 1 BIT, AND THE ABSOLUTE VALUE OF
C        THE MINIMUM SECOND ORDER DIFFERENCE INTO NBIT BITS.
C
      NBIT=1
 347  IF(ABS(IMIN).LT.2**NBIT)GO TO 348
      NBIT=NBIT+1
      GO TO 347
C
 348  CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,NBIT,5,IER,*900)
      IF(IMIN.LT.0)CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,1,1,IER,
     1                       *900)
      IF(IMIN.GE.0)CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,0,1,IER,
     1                       *900)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,ABS(IMIN),NBIT,IER,
     1          *900)
C
C        NOW PACK THE DATA VALUES THEMSELVES.
C
 3480 K=0
      NCOUNT=0
C
      DO 350 L=1,LX
C
      DO 349 M=1,NOV(L)
      K=K+1
      IF(LBIT(L).EQ.0)GO TO 349
C        GROUPS WITH ALL VALUES THE SAME ARE OMITTED.
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IC(K),LBIT(L),
     1            IER,*900)
      NCOUNT=NCOUNT+1
 349  CONTINUE
 350  CONTINUE
C
C        PAD WITH ZEROS TO AN EVEN OCTET.
C
      IFILL=MOD(33-IPOS,8)
      IF(IFILL.EQ.0)GO TO 360
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,IZERO,IFILL,IER,*900)
C        UPDATE OCTET 4 WITH UNUSUED BITS.
```

```
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC2,IPOS2,IFILL,4,IER,*900)

C        THE LENGTH OF SECTION 4 CAN NOW BE PUT INTO OCTETS 1-3.
C
 360  MOCTET=(LOC*4-(33-IPOS)/8)-(LOC1*4-(33-IPOS1)/8)
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC1,IPOS1,MOCTET,24,IER,*900)
C
C        FILL OCTETS 19 AND 20.
C
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC5,IPOS5,NCOUNT,16,IER,*900)
C
C        PACK END OF MESSAGE.
C
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC,IPOS,I7777,32,IER,*900)
C
C        FILL OCTETS 5-7 WITH THE TOTAL MESSAGE LENGTH IN OCTETS.
C
      IOCTET=LOC*4-(33-IPOS)/8
      CALL PKBG(KFIL10,KFIL12,IPACK,ND4,LOC0,IPOS0,IOCTET,24,IER,*900)
 370  CONTINUE
C
      RETURN
C
C        ERROR RETURN SECTION.
C
 900  WRITE(KFIL12,901)STATE,IER
 901  FORMAT('0ERROR IN GRIBC PACKING SECTION 'A4,'  IER ='I4)
      RETURN
      END
```

```
      SUBROUTINE GRIBPR(KFIL10,KFIL12,IC,IA,IB,IDIM,NXY,MINPK,
     1                  IMIN,IFIRST,IFOD,SECOND)
C
C        APRIL 1993   GLAHN   TDL   MICROVAX
C
C        PURPOSE
C            CALLED BY GRIBC TO DETERMINE WHETHER TO USE SECOND ORDER
C            DIFFERENCES OR ORIGINAL VALUES TO PACK.  ORIGINAL VALUES
C            ARE INDICATED WHEN THE AVERAGE RANGE OF CONSECUTIVE GROUPS
C            OF SIZE MINPK OF THE SECOND ORDER DIFFERENCES IS LARGER
C            THAN THE AVERAGE RANGE OF CONSECUTIVE GROUPS OF SIZE MINPK
C            OF THE ORIGINAL VALUES.
C
C        DATA SET USE
C           KFIL10 - UNIT NUMBER FOR CURRENT CONSOLE. (OUTPUT)
C           KFIL12 - UNIT NUMBER FOR OUTPUT (PRINT) FILE. (OUTPUT)
C
C        VARIABLES
C              KFIL10 = UNIT NUMBER FOR CURRENT CONSOLE.  (INPUT)
C              KFIL12 = UNIT NUMBER FOR OUTPUT (PRINT) FILE.  (INPUT)
C               IC(K) = HOLDS THE NXY ORIGINAL VALUES ON INPUT (K=1,IDIM).
C                       HOLDS THE NXY SECOND ORDER DIFFERENCES ON
C                       OUTPUT WHEN SECOND ORDER DIFFERENCES ARE TO BE
C                       USED.  IN THAT CASE, SECOND IS .TRUE. AND NXY
C                       HAS BEEN REDUCED BY 2.  (INPUT-OUTPUT)
C               IA(K) = WORK ARRAY (K=1,IDIM).  (INTERNAL)
C               IB(K) = WORK ARRAY (K=1,IDIM).  (INTERNAL)
C                IDIM = DIMENSION OF IC( ), IB( ), AND IA( ).  (INPUT)
C                 NXY = NUMBER OF VALUES IN IC( ) ON INPUT.  ON RETURN,
C                       NXY WILL ALSO BE THE NUMBER OF VALUES IN IC( ).
C                       IF THESE VALUES ARE SECOND ORDER DIFFERENCES,
C                       THEN NXY WILL HAVE BEEN REDUCED BY 2.  (INPUT-OUTPUT)
C              MINPK = INCREMENT IN WHICH RANGES WILL BE COMPUTED.  (INPUT)
C                IMIN = WHEN SECOND ORDER DIFFERENCES ARE USED, IMIN
C                       IS THE MINIMUM OF THEM.  (OUTPUT)
C              IFIRST = WHEN SECOND ORDER DIFFERENCES ARE USED, IFIRST
C                       IS THE FIRST ORIGINAL VALUE.  (OUTPUT)
C                IFOD = WHEN SECOND ORDER DIFFERENCES ARE USED, IFOD
C                       IS THE FIRST FIRST ORDER DIFFERENCE.  (OUTPUT)
C              SECOND = TRUE (FALSE) WHEN SECOND ORDER DIFFERENCES
C                       ARE (ARE NOT) USED.  (OUTPUT)
C
C        NON SYSTEM SUBROUTINES CALLED
C           NONE
C
      LOGICAL SECOND
C
      DIMENSION IC(IDIM),IA(IDIM),IB(IDIM)
C
C        COMPUTE FIRST ORDER DIFFERENCES.
C
      DO 120 K=1,NXY-1
      IA(K)=IC(K+1)-IC(K)
  120 CONTINUE
C
```

```
C         COMPUTE SECOND ORDER DIFFERENCES AND THEIR MIMIMUM.
C
      IMIN=999999
C
      DO 130 K=1,NXY-2
      IB(K)=IA(K+1)-IA(K)
      IF(IB(K).LT.IMIN)IMIN=IB(K)
 130  CONTINUE
C
C         COMPUTE AVERAGE RANGE OF NXY ORIGINAL VALUES IN INCREMENTS OF
C         MINPK.
C
      SUMR=0
      KOUNT=0
C
      DO 140 K=1,NXY,MINPK
      JMIN=999999
      JMAX=-999999
      IF(K+MINPK-1.GT.NXY)GO TO 140
C         THE LAST GROUP MAY BE VERY SMALL AND NOT BE REPRESENTATIVE OF
C         THE RANGE.
C
      DO 135 J=K,K+MINPK-1
      IF(IC(J).GT.JMAX)JMAX=IC(J)
      IF(IC(J).LT.JMIN)JMIN=IC(J)
 135  CONTINUE
C
      KOUNT=KOUNT+1
      IRANGE=JMAX-JMIN
      SUMR=SUMR+IRANGE
 140  CONTINUE
C
      AVGR=SUMR/KOUNT
C
C         COMPUTE AVERAGE RANGE OF NXY-2 2ND ORDER VALUES IN INCREMENTS OF
C         MINPK.
C
      SUMR=0
      KOUNT=0
C
      DO 150 K=1,NXY-2,MINPK
      JMIN=999999
      JMAX=-999999
      IF(K+MINPK-1.GT.NXY-2)GO TO 150
C         THE LAST GROUP MAY BE SMALL AND NOT REPRESENTATIVE OF THE RANGE.
C
      DO 145 J=K,K+MINPK-1
      IF(IB(J).GT.JMAX)JMAX=IB(J)
      IF(IB(J).LT.JMIN)JMIN=IB(J)
 145  CONTINUE
C
      KOUNT=KOUNT+1
      IRANGE=JMAX-JMIN
      SUMR=SUMR+IRANGE
 150  CONTINUE
```

```
C
      AVGR2=SUMR/KOUNT
C
      SECOND=.FALSE.
      WRITE(KFIL12,155)AVGR,AVGR2,IMIN
 155  FORMAT('0AVERAGE RANGE OF ORIGINAL SCALED VALUES =  'F10.2/
     1       ' AVERAGE RANGE OF SECOND ORDER DIFFERENCES ='F10.2/
     2       ' MINIMUM OF SECOND ORDER DIFFERENCES =    'I10)
      IF(AVGR2.GE.AVGR)GO TO 300
C
C        SECOND ORDER DIFFERENCES WILL BE PACKED.
C
      WRITE(KFIL12,160)
 160  FORMAT('0SECOND ORDER DIFFERENCES WILL BE PACKED')
C
      IFIRST=IC(1)
C
      DO 200 K=1,NXY-2
      IC(K)=IB(K)-IMIN
 200  CONTINUE
C
      IFOD=IA(1)
      SECOND=.TRUE.
      NXY=NXY-2
 300  RETURN
      END
```

```
      SUBROUTINE BGVRBL(KFIL10,KFIL12,IC,NDP,MINPK,INC,
     1                  JMIN,JMAX,IBIT,LBIT,NOV,NDQ,LX)
C
C         APRIL 1993   GLAHN   TDL   MICROVAX
C
C         PURPOSE
C             TO DETERMINE GROUPS OF VARIABLE SIZE, BUT AT LEAST OF
C             SIZE MINPK, AND THE ASSOCIATED MAX AND MIN OF EACH GROUP,
C             THE NUMBER OF BITS NECESSARY TO PACK EACH GROUP, AND THE
C             NUMBER OF VALUES IN EACH GROUP.  THE ROUTINE IS DESIGNED
C             TO DETERMINE THE GROUPS SUCH THAT A SMALL NUMBER OF BITS
C             IS NECESSARY TO PACK THE DATA WITHOUT EXCESSIVE
C             COMPUTATIONS.  IF ALL VALUES IN THE GROUP ARE ZERO, THE
C             NUMBER OF BITS TO USE IN PACKING IS DEFINED AS ZERO.
C             ALL VARIABLES ARE INTEGER.
C
C         DATA SET USE
C            KFIL10 - UNIT NUMBER FOR CURRENT CONSOLE. (OUTPUT)
C            KFIL12 - UNIT NUMBER FOR OUTPUT (PRINT) FILE. (OUTPUT)
C
C         VARIABLES IN CALL SEQUENCE
C               KFIL10 = UNIT NUMBER FOR CURRENT CONSOLE.  (INPUT)
C               KFIL12 = UNIT NUMBER FOR OUTPUT (PRINT) FILE.  (INPUT)
C                IC( ) = ARRAY TO HOLD DATA FOR PACKING.  THE VALUES
C                        DO NOT HAVE TO BE POSITIVE AT THIS POINT, BUT
C                        MUST BE IN THE RANGE -9999999 TO +9999999.
C                        THESE INTEGER VALUES WILL BE RETAINED EXACTLY
C                        THROUGH PACKING AND UNPACKING.  (INPUT)
C                  NDP = NUMBER OF VALUES IN IC( ).  ALSO TREATED
C                        AS ITS DIMENSION.  (INPUT)
C                MINPK = THE MINIMUM SIZE OF EACH GROUP, EXCEPT POSSIBLY
C                        THE LAST ONE.  (INPUT)
C                  INC = THE NUMBER OF VALUES TO ADD TO AN ALREADY
C                        EXISTING GROUP IN DETERMINING WHETHER OR NOT
C                        TO START A NEW GROUP.  IDEALLY, THIS WOULD BE
C                        1, BUT EACH TIME INC VALUES ARE ATTEMPTED, THE
C                        MAX AND MIN OF THE NEXT MINPK VALUES MUST BE
C                        FOUND.  THIS IS "A LOOP WITHIN A LOOP," AND
C                        A SLIGHTLY LARGER VALUE MAY GIVE ABOUT AS GOOD
C                        RESULTS WITH SLIGHTLY LESS COMPUTATIONAL TIME.
C                        IF INC IS LE 0, 1 IS USED, AND A DIAGNOSTIC IS
C                        OUTPUT.  (INPUT)
C              JMIN(J) = THE MINIMUM OF EACH GROUP (J=1,LX).  (OUTPUT)
C              JMAX(J) = THE MAXIMUM OF EACH GROUP (J=1,LX).  (OUTPUT)
C                 IBIT = THE NUMBER OF BITS NECESSARY TO PACK JMIN(J)
C                        VALUES, J=1,LX.  (OUTPUT)
C              LBIT(J) = THE NUMBER OF BITS NECESSARY TO PACK EACH GROUP
C                        (J=1,LX).  IT IS ASSUMED THE MINIMUM OF EACH
C                        GROUP WILL BE REMOVED BEFORE PACKING, AND THE
C                        VALUES TO PACK WILL, THEREFORE, ALL BE POSITIVE.
C                        HOWEVER, IC( ) DOES NOT NECESSARILY CONTAIN
C                        ALL POSITIVE VALUES.  IF THE OVERALL MINIMUM
C                        HAS BEEN REMOVED, THEN IC( ) WILL CONTAIN
C                        ONLY POSITIVE VALUES.  (OUTPUT)
C               NOV(J) = THE NUMBER OF VALUES IN EACH GROUP (J=1,LX).
```

```
C                    (OUTPUT)
C                NDQ = THE DIMENSION OF JMIN( ), JMAX( ), LBIT( ), AND
C                      NOV( ).  (INPUT)
C                 LX = THE NUMBER OF GROUPS DETERMINED.  (OUTPUT)
C
C        INTERNAL VARIABLES
C                KINC = WORKING COPY OF INC.  MAY BE MODIFIED.
C                MINA = MINUMUM VALUE IN GROUP A.
C                MAXA = MAXIMUM VALUE IN GROUP A.
C               IBITA = NUMBER OF BITS NEEDED TO HOLD VALUES IN GROUP A.
C                MINB = MINUMUM VALUE IN GROUP B.
C                MAXB = MAXIMUM VALUE IN GROUP B.
C               IBITB = NUMBER OF BITS NEEDED TO HOLD VALUES IN GROUP B.
C                MINC = MINUMUM VALUE IN GROUP C.
C                MAXC = MAXIMUM VALUE IN GROUP C.
C              KTOTAL = COUNT OF NUMBER OF VALUES IN IC( ) PROCESSED.
C               NOUNT = NUMBER OF VALUES ADDED TO GROUP A TO MAKE GROUP .
C
C        NON SYSTEM SUBROUTINES CALLED
C           NONE
C
      DIMENSION IC(NDP)
      DIMENSION JMIN(NDQ),JMAX(NDQ),LBIT(NDQ),NOV(NDQ)
C
      IF(INC.LE.0)WRITE(KFIL12,100)INC
  100 FORMAT('0INC ='I8,' NOT CORRECT.  1 IS USED.')
      KINC=MAX(INC,1)
      KSTART=1
      KTOTAL=0
      LX=0
  110 KOUNTA=0
      IBITA=0
      MINA=9999999
      MAXA=-9999999
C
C        FIND THE MIN AND MAX OF GROUP A.  THIS WILL INITALLY BE OF
C        SIZE MINPK (IF THERE ARE STILL MINPK VALUES IN IC( )), BUT
C        WILL INCREASE IN SIZE IN INCREMENTS OF INC UNTIL A NEW
C        GROUP IS STARTED.
C
      NEND=MIN(KSTART+MINPK-1,NDP)
      IF(NDP-NEND.LE.MINPK/2)NEND=NDP
C
      DO 120 K=KSTART,NEND
      MINA=MIN(MINA,IC(K))
      MAXA=MAX(MAXA,IC(K))
      KOUNTA=KOUNTA+1
  120 CONTINUE
C
C        INCREMENT KTOTAL AND FIND THE BITS NEEDED TO PACK THE A GROUP.
C
      KTOTAL=KTOTAL+KOUNTA
  125 IF(MAXA-MINA.LT.2**IBITA)GO TO 130
      IBITA=IBITA+1
      GO TO 125
```

```
C
 130  IF(KTOTAL.GE.NDP)GO TO 200
C
C        MORE VALUES LEFT IN IC( ).  TRY TO ADD INC VALUES TO GROUP A.
C        THIS AUGMENTED GROUP IS CALLED GROUP C.
C
      MINC=MINA
      MAXC=MAXA
 135  NOUNT=0
      IF(NDP-(KTOTAL+INC).LE.MINPK/2)KINC=NDP-KTOTAL
C        ABOVE STATEMENT CONSTRAINS THE LAST GROUP TO BE NOT LESS THAN
C        MINPK/2 IN SIZE.  IF A PROVISION LIKE THIS IS NOT INCLUDED,
C        THERE WILL ALMOST ALWAYS BE A VERY SMALL GROUP AT THE END.
C
      DO 140 K=KTOTAL+1,MIN(KTOTAL+KINC,NDP)
      MINC=MIN(MINC,IC(K))
      MAXC=MAX(MAXC,IC(K))
      NOUNT=NOUNT+1
 140  CONTINUE
C
C        IF THE NUMBER OF BITS NEEDED FOR GROUP C IS GT IBITA,
C        THEN THIS GROUP A IS A GROUP TO PACK.
      IF(MAXC-MINC.GE.2**IBITA) GO TO 200
C
C        THE BITS NECESSARY FOR GROUP C HAS NOT INCREASED FROM THE
C        BITS NECESSARY FOR GROUP A.  FIND PACKING BITS OVER MINPK VALUES
C        FOLLOWING GROUP A, THAT IS GROUP B.
C
      MINB=9999999
      MAXB=-9999999
      IBITB=0
      JOUNT=0
C
      DO 160 K=KTOTAL+1,MIN(KTOTAL+MINPK,NDP)
      MINB=MIN(MINB,IC(K))
      MAXB=MAX(MAXB,IC(K))
      JOUNT=JOUNT+1
 160  CONTINUE
C
 165  IF(MAXB-MINB.LT.2**IBITB)GO TO 170
      IBITB=IBITB+1
      GO TO 165
C
C        DETERMINE WHETHER THE NEXT MINPK VALUES CAN BE PACKED IN
C        LESS BITS THAN GROUP A.  IF SO, PACK GROUP A AND START
C        ANOTHER GROUP.
C
 170  IF(IBITB.LT.IBITA)GO TO 200
C
C        IBITB GE IBITA.  THEREFORE, ADD THIS INCREMENT TO A.
C
      KTOTAL=KTOTAL+NOUNT
      KOUNTA=KOUNTA+NOUNT
      MINA=MINC
      MAXA=MAXC
```

```
C        KOUNTA IS THE NUMBER OF VALUES IN GROUP A.  THIS GROUP WILL
C        NEVER BE SPLIT.
      IF(KTOTAL.LT.NDP)GO TO 135
C
C        GROUP A IS TO BE PACKED.  STORE VALUES IN JMIN( ), JMAX( ),
C        LBIT( ), AND NOV( ).
C
 200  LX=LX+1
      IF(LX.LE.NDQ)GO TO 205
      WRITE(KFIL12,201)
 201  FORMAT('0LX NOT LARGE ENOUGH.  STOP IN VRBLPK AT 201')
      STOP 201
C
 205  JMIN(LX)=MINA
      JMAX(LX)=MAXA
      LBIT(LX)=IBITA
      NOV(LX)=KOUNTA
      KSTART=KTOTAL+1
      IF(KTOTAL.LT.NDP)GO TO 110
C        WITH THE ABOVE TRANSFER, A NEW GROUP A OF SIZE MINPK WILL
C        BE DEFINED.
C
C        CALUCLATE IBIT, THE NUMBER OF BITS NEEDED TO HOLD THE GROUP
C        MINIMUN VALUES.
C
      IBIT=1
C
 210  DO 220 L=1,LX
      IF(JMIN(L).LT.2**IBIT)GO TO 220
      IBIT=IBIT+1
      GO TO 210
C
 220  CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE PKBG(KFIL10,KFIL12,IPACK,NDX,LOC,IPOS,NVALUE,NBIT,
     1                IER,*)
C
C         APRIL 1993    GLAHN    TDL    MICROVAX
C
C         PURPOSE
C             PACKS NBIT BITS IN THE POSITIVE INTEGER NVALUE INTO ARRAY
C             IPACK(NDX) STARTING IN WORD LOC, BIT IPOS.  THE WORD
C             POINTER LOC AND BIT POSITION POINTER IPOS ARE UPDATED
C             AS NECESSARY.  PACKING WILL NOT OCCUR IF IPACK( ) WOULD
C             BE OVERFLOWED.  IN THAT CASE, RETURN IS WITH IER=1
C             RATHER THAN FOR THE GOOD RETURN IER=0.  WHEN NBIT EQ 0
C             AND NVALUE EQ 0, NO PACKING IS DONE.  THIS ROUTINE ACTS
C             AS "INSERTION" RATHER THAN "ADDITION."  THAT IS, THE
C             BITS, IF ANY, TO THE RIGHT OF THE PACKED VALUE
C             ARE RETAINED.  THIS MEANS THAT THE IPACK( ) ARRAY SHOULD
C             BE ZEROED OUT BEFORE USING.  ALSO, ANY INSERTION MUST BE
C             BE INTO AN AREA THAT HAS ALL ZERO BITS.
C
C         DATA SET USE
C            KFIL10 - UNIT NUMBER FOR CURRENT CONSOLE. (OUTPUT)
C            KFIL12 - UNIT NUMBER FOR OUTPUT (PRINT) FILE. (OUTPUT)
C
C         VARIABLES
C               KFIL10 = UNIT NUMBER FOR CURRENT CONSOLE.  (INPUT)
C               KFIL12 = UNIT NUMBER FOR OUTPUT (PRINT) FILE.  (INPUT)
C             IPACK(J) = ARRAY TO PACK INTO (J=1,NDX).  (INPUT-OUTPUT)
C                  NDX = DIMENSION OF IPACK( ).  (INPUT)
C                  LOC = WORD IN IPACK( ) TO START PACKING.  UPDATED
C                        AS NECESSARY AFTER PACKING IS COMPLETED.
C                        (INPUT-OUTPUT)
C                 IPOS = BIT POSITION (COUNTING LEFTMOST BIT IN WORD
C                        AS 1) TO START PACKING.  MUST BE GE 1 AND
C                        LE 32.  UPDATED AS NECESSARY
C                        AFTER PACKING IS COMPLETED.  (INPUT-OUTPUT)
C               NVALUE = THE RIGHTMOST NBIT BITS IN NVALUE WILL
C                        BE PACKED.  (INPUT)
C                 NBIT = SEE NVALUE.  MUST BE GE 0 AND LE 32.  (INPUT)
C                  IER = STATUS RETURN:
C                        0 = GOOD RETURN.
C                        1 = PACKING WOULD OVERFLOW IPACK( ).
C                        2 = IPOS NOT IN RANGE 1 TO 32.
C                        3 = NBIT NOT IN RANGE 0 TO 32.
C                        4 = NBIT EQ 0, BUT NVALUE NE 0.
C                      * = ALTERNATE RETURN WHEN IER NE 0.
C
C         NON SYSTEM SUBROUTINES CALLED
C             NONE (VAX FORTRAN 77 EXTENSIONS ISHFT AND IOR ARE USED)
C
      DIMENSION IPACK(NDX)
C
```

```
C        CHECK CORRECTNESS OF INPUT AND SET STATUS RETURN.
C
      IER=0
      IF(NBIT.EQ.0.AND.NVALUE.EQ.0)GO TO 150
      IF(NBIT.NE.0)GO TO 111
      IER=4
C        WHEN NBIT=0, NVALUE MUST BE ALSO.
 111  IF(LOC+(IPOS+NBIT-2)/32.GT.NDX)IER=1
C        PACKING WOULD OVERFLOW IPACK( ).
      IF(IPOS.LE.0.OR.IPOS.GT.32)IER=2
      IF(NBIT.LT.0.OR.NBIT.GT.32)IER=3
      IF(IER.NE.0)RETURN 1
C
C        SHIFT KDONOR=NVALUE TO LEFT TO ELIMINATE UNWANTED BITS, IF ANY,
C        AND BACK TO RIGHT TO MATCH LOCATION IN IPACK(LOC).
C
      KDONOR=NVALUE
      KDONOR=ISHFT(ISHFT(NVALUE,32-NBIT),1-IPOS)
      IPACK(LOC)=IOR(IPACK(LOC),KDONOR)
C        PACKING COMPLETE UNLESS NOT ALL NBIT BITS WOULD FIT INTO
C        WORD IPACK(LOC).
      IF(IPOS+NBIT.GT.33)GO TO 140
C
C        PACKING COMPLETE.  UPDATE LOC AND IPOS AS NECESSARY.
C
      LOC=LOC+(IPOS-1+NBIT)/32
      IPOS=MOD(IPOS-1+NBIT,32)+1

      GO TO 150
C
C        MUST COMPLETE PACKING.
C
 140  IPOS=IPOS+NBIT-32
      LOC=LOC+1
      IPACK(LOC)=IOR(ISHFT(NVALUE,33-IPOS),IPACK(LOC))
C
 150  CONTINUE
      RETURN
      END
```

APPENDIX II

This appendix contains a listing of the unpacking FORTRAN code to implement the options recommended in this office note. It consists of two routines, DEGRIBC and UNPKBG, the latter being called by the former. Another routine called by DEGRIBC, Q9IV32, is for use only when implementing this code on a VAX computer, and is not included.

```
      SUBROUTINE DEGRIBC(KFIL10,KFIL12,A,IA,IC,IPACK,ND1,IS0,IS1,IS2,
     1                   IS4,ND8,LBIT,JMIN,NOV,ND4,KOCTET,IER)
C
C       APRIL 1993   GLAHN   TDL   MICROVAX
C
C       PURPOSE
C           SUBROUTINE TO UNPACK (DEGRIB) DATA PACKED BY ROUTINE
C           GRIBC; ONLY THE OPTIONS IN GRIBC ARE DEALT WITH IN
C           DEGRIBC.  SEE GRIBC AND TDL OFFICE NOTE 93-XX FOR
C           FURTHER INFORMATION.
C
C       DATA SET USE
C          KFIL10 - UNIT NUMBER FOR CURRENT CONSOLE. (OUTPUT)
C          KFIL12 - UNIT NUMBER FOR OUTPUT (PRINT) FILE. (OUTPUT)
C
C       VARIABLES
C             KFIL10 = UNIT NUMBER FOR CURRENT CONSOLE.  (INPUT)
C             KFIL12 = UNIT NUMBER FOR OUTPUT (PRINT) FILE.  (INPUT)
C               A(K) = UNPACKED DATA RETURNED (K=1,ND1).  THE DATA ARE IN
C                      THIS LINEAR ARRAY, IN THE ORDER OF A FORTRAN
C                      2-DIMENSIONAL ARRAY LIKE A(IX,JY) (IX=1,NX) (JY=1,NY).
C                      (OUTPUT)
C              IA(K) = WORK ARRAY (K=1,ND1).  HOLDS UNPACKED VALUES.
C                      (INTERNAL)
C              IC(K) = WORK ARRAY (K=1,ND1).  (INTERNAL)
C           IPACK(J) = THE ARRAY HOLDING THE ACTUAL PACKED MESSAGE
C                      (J=1,MAX OF ND1).  (INPUT)
C                ND1 = DIMENSION OF A( ), IA( ), IC( ), AND IPACK( ).
C             IS0(L) = HOLDS THE OCTETS DECODED FROM SECTION 0 (L=1,ND8).
C                      NOT ALL LOCATIONS ARE USED.  L CORRESPONDS TO THE
C                      (BEGINNING) OCTET OF THE IDENTIFIERS DEFINED
C                      FOR THE GRIB MESSAGE.  (OUTPUT)
C             IS1(L) = HOLDS THE OCTETS DECODED FROM SECTION 1 (L=1,ND8).
C                      NOT ALL LOCATIONS ARE USED.  L CORRESPONDS TO THE
C                      (BEGINNING) OCTET OF THE IDENTIFIERS DEFINED
C                      FOR THE GRIB MESSAGE.  (OUTPUT)
C             IS2(L) = HOLDS THE OCTETS DECODED FROM SECTION 2 (L=1,ND8).
C                      NOT ALL LOCATIONS ARE USED.  L CORRESPONDS TO THE
C                      (BEGINNING) OCTET OF THE IDENTIFIERS DEFINED
C                      FOR THE GRIB MESSAGE.  (OUTPUT)
C             IS4(L) = HOLDS THE OCTETS DECODED FROM SECTION 4 (L=1,ND8).
C                      NOT ALL LOCATIONS ARE USED.  L CORRESPONDS TO THE
C                      (BEGINNING) OCTET OF THE IDENTIFIERS DEFINED
C                      FOR THE GRIB MESSAGE.  (OUTPUT)
C                ND8 = DIMENSION OF IS0( ), IS1( ), IS2( ),AND IS4.  (INPUT)
C             LBIT(M) = THE NUMBER OF BITS NECESSARY TO HOLD THE
C                      PACKED VALUES FOR EACH GROUP M (M=1,LX).  (OUTPUT)
```

```
C            JMIN(M) = THE MINIMUM VALUE SUBTRACTED FOR EACH GROUP
C                      M BEFORE PACKING (M=1,LX).  (OUTPUT)
C             NOV(M) = THE NUMBER OF VALUES IN GROUP M (M=1,LX).
C                      (OUTPUT)
C                ND4 = THE SIZE OF THE ARRAYS LBIT( ), NOV( ), AND
C                      JMIN( ).  (INPUT)
C             KOCTET = THE TOTAL MESSAGE SIZE IN OCTETS = IS0(5).  (OUTPUT)
C                IER = ERROR RETURN.  MOST ERROR RETURNS NE 0 ARE
C                      FROM SUBROUTINE UNPKBG OR Q9IV32.  OTHERS USED ARE:
C                      IER = 10--CAN'T FIND BEGINNING OF MESSAGE.
C                      IER = 11--CAN'T FIND END OF MESSAGE.
C                      IER = 20--IS1(8) IS INCORRECT.  MUST BE 128 DECIMAL.
C                      IER = 21--IS2(6) IS INCORRECT.  MUST BE 3 OR 5.
C                      IER = 22--IS2(28) IS INCORRECT.  MUST BE 4 OR 5.
C                      IER = 23--IS4(4) OR IS4(14) IS INCORRECT.  IS4(4) MUST
C                            BE 7 AND IS4(14) MUST BE 48.
C                 LX = THE NUMBER OF VALUES IN LBIT( ), JMIN( ), AND
C                      NOV( ).  TAKEN FROM IS4(17).  (INTERNAL)
C              SCALE = SCALING PARAMETER = 10.**(-IS1(27)).  (INTERNAL)
C               MINA = THE MINIMUM VALUE THAT WAS SUBTRACTED BEFORE PACKING
C                      (I.E., THE REFERENCE VALUE).  NOTE THAT ALTHOUGH
C                      THE REFERENCE VALUE IS FLOATING POINT (REAL),
C                      IT IS ASSUMED TO BE A WHOLE NUMBER.  (INTERNAL)
C             NVALUE = AN UNPACKED VALUE RETURNED FROM SUBROUTINE
C                      UNPKBG.  EQUIVALENCED TO CVALUE.
C              C7777 = EQUIVALENCED TO I7777 FOR TESTING.  (INTERNAL)
C                      (CHARACTER*4)
C              I7777 = EQUIVALENCED TO C7777 FOR TESTING.  (INTERNAL)
C              IGRIB = SET TO IS0(1), WHICH SHOULD BE 'GRIB'.  (INTERNAL)
C              CGRIB = EQUIVALENCED TO IGRIB FOR TESTING.  (INTERNAL)
C                      (CHARACTER*4)
C               MREF = SET TO IS4(7), WHICH IS ACTUALLY A FLOATING POINT
C                      NUMBER.  EQUIVALENCED TO REF.  (INTERNAL)
C                REF = EQUIVALENCED TO MREF, SO THAT IT CAN BE USED
C                      AS A FLOATING POINT NUMBER WITHOUT GETTING IT
C                      CONVERTED.  (INTERNAL)
C             MOCTET = HOLDS THE NUMBER OF OCTETS AT VARIOUS STEPS OF
C                      UNPACKING FROM WHICH LOC AND IPOS ARE
C                      DETERMINED.  (INTERNAL)
C                LOC = HOLDS WORD POSITION IN IPACK OF NEXT VALUE TO
C                      UNPACK.  (INTERNAL)
C               IPOS = HOLDS BIT POSITION IN IPACK(LOC) OF THE FIRST
C                      BIT OF THE NEXT VALUE TO UNPACK.
C               KBIT = THE NUMBER OF BITS TO HOLD THE WIDTHS OF WIDTHS
C                      FROM OCTET 22 (THE SIZE OF THE VALUES PUT INTO
C                      LBIT( ).
C               IBIT = THE NUMBER OF BITS REQUIRED TO PACK THE
C                      FIRST VALUE IN THE FIELD (WITH THE REFERENCE
C                      REMOVED).  USED ONLY WHEN PACKING SECOND ORDER
C                      DIFFERENCES.
C               JBIT = THE NUMBER OF BITS REQUIRED TO PACK THE
C                      ABSOLUTE VALUE OF THE FIRST FIRST ORDER
C                      DIFFERENCE.  USED ONLY WHEN PACKING SECOND ORDER
C                      DIFFERENCES.
```

```
C               NBIT = THE NUMBER OF BITS REQUIRED TO PACK THE
C                      ABSOLUTE VALUE OF THE MINIMUM SECOND ORDER
C                      DIFFERENCE.  USED ONLY WHEN PACKING SECOND ORDER
C                      DIFFERENCES.
C             IFIRST = THE FIRST VALUE IN THE FIELD (WITH THE
C                      REFERENCE REMOVED).  USED ONLY WHEN PACKING SECOND
C                      ORDER DIFFERENCES.
C               IFOD = THE FIRST FIRST ORDER DIFFERENCE.  USED ONLY WHEN
C                      PACKING SECOND ORDER DIFFERENCES.
C              STATE = HOLDS 4 CHARACTERS FOR PRINTOUT IN CASE OF ERROR.
C                      (INTERNAL)  (CHARACTER*4)
C
C        NON SYSTEM SUBROUTINES CALLED
C           UNPKBG, Q9IV32
C
      CHARACTER*4 C7777/'7777'/,
     1            CGRIB,
     2            STATE
      LOGICAL SECOND
C
      DIMENSION A(ND1),IA(ND1),IC(ND1),IPACK(ND1)
      DIMENSION JMIN(ND4),NOV(ND4),LBIT(ND4)
      DIMENSION IS0(ND8),IS1(ND8),IS2(ND8),IS4(ND8)
      EQUIVALENCE (CGRIB,IGRIB),(C7777,I7777),(MREF,REF)
C
C        SET ERROR RETURN AND ZERO ARRAYS.
C
      IER=0
C
      DO 110 K=1,ND8
      IS0(K)=0
      IS1(K)=0
      IS2(K)=0
      IS4(K)=0
 110  CONTINUE
C
C        UNPACK SECTION 0.
C
      STATE='0   '
      LOC=1
      IPOS=1
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS0(1),32,IER,*900)
      IGRIB=IS0(1)
      IF(CGRIB.EQ.'GRIB')GO TO 112
      WRITE(KFIL12,111)CGRIB
 111  FORMAT('0BEGINNING OF GRIB MESSAGE NOT FOUND.'
     1       '  FIRST 4 CHARACTERS =  'A4)
      IER=10
      GO TO 380
C
 112  CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS0(5),24,IER,*900)
      KOCTET=IS0(5)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS0(8),8,IER,*900)
C
```

```
C        UNPACK SECTION 1.
C
      STATE='1   '
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS1(1),24,IER,*900)
C
      DO 210 K=4,10
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS1(K),8,IER,*900)
 210  CONTINUE
C
      IF(IS1(8).EQ.128)GO TO 215
C        IS1(8) IS NOT OF CORRECT VALUE.  IT MUST BE 128 DECIMAL.
      WRITE(KFIL12,212)IS1(8)
 212  FORMAT('0IS1(8) ='I6,' IS INCORRECT.')
      IER=20
      GO TO 380
C
 215  CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS1(11),16,IER,*900)
C
      DO 220 K=13,21
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS1(K),8,IER,*900)
 220  CONTINUE
C
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS1(22),16,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS1(24),8,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS1(25),8,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS1(26),8,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,ISIGN,1,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS1(27),15,IER,*900)
      IF(ISIGN.EQ.1)IS1(27)=-IS1(27)
C        A NEGATIVE DECIMAL SCALE FACTOR MUST HAVE THE MINUS SIGN,
C        IF PRESENT, IN BIT 1 OF THE 16-BIT FIELD.
C
C        UNPACK SECTION 2.
C
C        START ACCORDING TO LENGHT OF SECTION 0 = 8 OCTETS AND
C        SECTION 1 = IS1(1) OCTETS.
C
      STATE='2   '
      MOCTET=8+IS1(1)
      LOC=MOCTET/4+1
      IPOS=(MOCTET-(LOC-1)*4)*8+1
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(1),24,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(4),8,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(5),8,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(6),8,IER,*900)
      IF(IS2(6).EQ.3.OR.IS2(6).EQ.5)GO TO 228
      WRITE(KFIL12,227)IS2(6)
 227  FORMAT('0IS2(6) ='I6,'  INDICATES NEITHER POLAR STEREOGRAPHIC',
     1       ' NOR LAMPBERT MAP.')
      IER=21
      GO TO 380
C
 228  CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(7),16,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(9),16,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(11),24,IER,*900)
```

```
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(14),24,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(17),8,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(18),24,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(21),24,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(24),24,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(27),8,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(28),8,IER,*900)
      IF(IS2(6).NE.5)GO TO 230
C        THIS IS POLAR STEREOGRAPHIC
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(29),32,IER,*900)
      GO TO 300
C        THIS IS LAMBERT.  IS2(6) CHECKED ABOVE.
 230  CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(29),24,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(32),24,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(35),24,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(38),24,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS2(41),16,IER,*900)
 300  CONTINUE
C
C        UNPACK SECTION 3.
      IS3=0
C        NOT IMPLEMENTED.
C
C        UNPACK SECTION 4.
C
C        START ACCORDING TO LENGTH OF SECTION 0 = 8 OCTETS PLUS
C        SECTION 1 = IS2(1) OCTETS PLUS SECTION 2 = IS2(1) IOCTETS.
C
      STATE='4   '
      MOCTET=MOCTET+IS2(1)
C        MOCTET = THE OCTETS BEFORE SECTION 4.
      LOC=MOCTET/4+1
      IPOS=(MOCTET-(LOC-1)*4)*8+1
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS4(1),24,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS4(4),4,IER,*900)
C        ONLY THE LEFTMOST 4 BITS OF OCTET 4 ARE NEEDED.
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,ISX,4,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS4(5),16,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS4(7),32,IER,*900)
C        IS4(7) IS FLOATING POINT.
      MREF=IS4(7)
C*********************
C        NOTE:  THE FLOATING POINT VALUE PACKED, AND UNPACKED INTO
C        IS4(7), IS THE GRIB IBM REPRESENTATION.  IF THIS UKPACKING
C        PROGRAM IS RUN ON A VAX MACHINE, THEN INSERT THE FOLLOWING
C        THREE STATEMENTS TO CONVERT TO VAX FLOATING POINT.
      CALL Q9IV32(IS4(7),REF,1,ISTAT)
      IF(ISTAT.EQ.0)GO TO 318
      WRITE(KFIL10,317)ISTAT
 317  FORMAT('0CONVERSION PROBLEM IN Q9IV32, CALLED FROM DEGRIG. ',
     1       ' ISTAT ='I10)
      IER=ISTAT
      GO TO 380
C*********************
```

```
 318  MINA=REF
      IS4(7)=MINA
C        ABOVE CONVERTS FLOATING POINT VALUE TO INTEGER.  THIS ASSUMES
C        THE PACKING HAS BEEN DONE TO UNITS AFTER USING THE 10'S
C        EXPONENT.  THAT IS, NO MORE RESOLUTIN IS RETAINED THAN UNTIS
C        AFTER SCALING BY THE POWER OF 10.  IS4(7) IS SET TO THAT VALUE
C        FOR RETURN TO CALLING PROGRAM.
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS4(11),8,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS4(12),16,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS4(14),8,IER,*900)
      SECOND=.FALSE.
      IF(ISHFT(ISHFT(IS4(14),28),-31).NE.0)SECOND=.TRUE.
C        A 1 IN BIT POSITION 5 OF OCTET IS4(14) (BIT POSITION 29 OF THE
C        FULL WORD) INDICATES THAT SECOND ORDER DIFFERENCES ARE PACKED.
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS4(15),16,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS4(17),16,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS4(19),16,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IS4(21),8,IER,*900)
C
      IF(IS4(4).EQ.7.AND.IS4(14).EQ.48)GO TO 323
C        THE ONLY UNPACKING ACCOMMODATED IS REPRESENTED BY
C        IS4(4) =   7 DECIMAL =  7 OCTAL =    111 BINARY, AND
C        IS4(14) = 48 DECIMAL = 60 OCTAL = 110000 BINARY.
      WRITE(KFIL12,322)IS4(4),IS4(14)
 322  FORMAT('0INCORRECT PACKING INFORMATION. '
     1       ' IS4(4) ='I6,'  IS4(14) ='I6)
      IER=23
      GO TO 380
C
C        RETRIEVE IS4(17) = LX WIDTHS IN LBIT( ), BUT FIRST RETRIEVE
C        THE NUMBER OF BITS USED TO HOLDTHE WIDTH OF WIDTHS.
C
 323  LX=IS4(17)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,KBIT,8,IER,*900)
C
      DO 325 K=1,LX
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,LBIT(K),KBIT,IER,
     1            *900)
 325  CONTINUE
C
C        RETRIEVE BITS NEEDED FOR GROUP SIZES AND THE GROUP SIZES.
C
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,MBIT,8,IER,*900)
C
      DO 330 L=1,LX
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,NOV(L),MBIT,IER,*900)
 330  CONTINUE
C
C        RETRIEVE LX 2ND ORDER MINIMA.  START ACCORDING TO IS4(12) OCTETS.
C
      IOCTET=IS4(12)-1+(8+IS1(1)+IS2(1)+IS3)
C        IOCTET = OCTETS BEFORE START OF GROUP MINIMA, COUNTING
C        FROM THE START OF THE MESSAGE.
      LOC=IOCTET/4+1
      IPOS=(IOCTET-(LOC-1)*4)*8+1
```

```
C
      DO 335 L=1,LX
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,JMIN(L),IS4(11),
     1           IER,*900)
 335  CONTINUE
C
C        RETIREVE PACKED VALUES, AND RESTORE ORIGINAL VALUES.
C        START ACCORDING TO IS4(15) OCTET.
C
      IOCTET=IS4(15)-1+(8+IS1(1)+IS2(1)+IS3)
C        IOCTET = OCTETS BEFORE START OF 2ND ORDER VALUES, COUNTING
C        FROM THE START OF THE MESSAGE.
      LOC=IOCTET/4+1
      IPOS=(IOCTET-(LOC-1)*4)*8+1
      IF(.NOT.SECOND)GO TO 337
C
C        SECOND ORDER DIFFERENCES ARE PACKED; OTHER INFORMATION MUST BE
C        RETRIEVED HERE.
C
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IBIT,5,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IFIRST,IBIT,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,JBIT,5,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,ISIGN,1,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IFOD,JBIT,IER,*900)
      IF(ISIGN.GT.0)IFOD=-IFOD
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,NBIT,5,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,ISIGN,1,IER,*900)
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,IMIN,NBIT,IER,*900)
      IF(ISIGN.GT.0)IMIN=-IMIN
C
C        RETRIEVE PACKED VALUES AND RESTORE ORIGINAL VALUES.  WHEN
C        SECOND ORDER DIFFERENCES ARE PACKED, THOSE DIFFERENCES WILL
C        BE IN IA( ) RATHER THAN THE ORIGINAL VALUES.
C
 337  K=0
C
      DO 340 L=1,LX
      NVALUE=0
C
      DO 339 M=1,NOV(L)
      IF(LBIT(L).EQ.0)GO TO 338
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,NVALUE,LBIT(L),IER,
     1           *900)
 338  K=K+1
      IA(K)=JMIN(L)+NVALUE
 339  CONTINUE
 340  CONTINUE
C
      NXY=IS2(7)*IS2(9)
      NX=IS2(7)
      NY=IS2(9)
C
```

```
C        UNPACK END OF MESSAGE FOR CHECK.
C        START ACCORDING TO TOTAL LENGTH OF MESSAGE.
C
      STATE='END '
      IOCTET=IS0(5)-4
      LOC=IOCTET/4+1
      IPOS=(IOCTET-(LOC-1)*4)*8+1
      CALL UNPKBG(KFIL10,KFIL12,IPACK,ND1,LOC,IPOS,NVALUE,32,IER,*900)
      IF(NVALUE.EQ.I7777)GO TO 3440
      WRITE(KFIL12,344)NVALUE
 344  FORMAT('0END OF MESSAGE NOT FOUND.  4 CHARACTERS FOUND ARE  ',O15)
      IER=11
      GO TO 380
C
 3440 IF(SECOND)GO TO 3444
C
C        SECOND ORDER DIFFERENCES NOT USED.  PUT OVERALL MINIMUM
C        BACK IN.
C
      DO 3442 K=1,NXY
      IA(K)=IA(K)+MINA
 3442 CONTINUE
C
C        SECOND ORDER DIFFERENCES ARE USED, THE NXY-2 VALUES BEING
C        IN IC( ).  IFIRST IS THE FIRST (ORIGINAL) VALUE, AND IFOD
C        IS THE FIRST FIRST ORDER DIFFERENCE.  NOW RESTORE ORIGINAL
C        VALUES.
C
 3444 DO 345 K=1,NXY-2
      IPACK(K)=IA(K)+IMIN
 345  CONTINUE
C
      ISUM=0
      ISUM2=0
      IA(1)=IFIRST+MINA
      IA(2)=IA(1)+IFOD
C
      DO 347 K=3,NXY
      ISUM2=ISUM2+IPACK(K-2)
      IC(K)=ISUM2+IFOD
      ISUM=ISUM+IC(K)
      IA(K)=ISUM+IA(2)
 347  CONTINUE
C
C        APPLY SCALE FACTOR FOR "NORMAL" SEQUENCE OF POINTS.
C
 360  SCALE=10.**(-IS1(27))
      IF(IS2(28).EQ.4)GO TO 362
C        IS2(28) MUST BE EITHER 4, INDICATING "NORMAL" SEQUENCE OF
C        POINTS, OR 5, INDICATING A REVERSAL OF ALTERNATE ROWS TO
C        FACILITATE EFFICEINCY IN 2ND ORDER PACKING SCHEME.
      IF(IS2(28).EQ.5)GO TO 367
      WRITE(KFIL12,361)IS2(28)
 361  FORMAT('0IS2(28) DOES NOT INDICATE A SEQUENCE OF POINTS THAT',
     1       ' THIS DECODER CAN HANDLE.')
```

```
      IER=22
      GO TO 380
C
 362  DO 365 K=1,NXY
      A(K)=IA(K)*SCALE
 365  CONTINUE
C
      GO TO 380
C
C        REVERSE ALTERNATE ROWS IF PACKED THAT WAY AND APPLY SCALE FACTOR.
C
 367  K=0
C
      DO 370 JY=1,NY
      DO 369 IX=1,NX
      K=K+1
C
        IF(MOD(JY,2).EQ.0)THEN
           M=JY*NX+1-IX
        ELSE
           M=(JY-1)*NX+IX
        END IF
C
      A(K)=IA(M)*SCALE
 369  CONTINUE
 370  CONTINUE
C
 380  RETURN
C
C        ERROR RETURN SECTION.
C
 900  WRITE(KFIL12,901)STATE,IER
 901  FORMAT('0ERROR IN DEGRIBC SECTION 'A4,'  IER ='I4)
      RETURN
      END
```

```
      SUBROUTINE UNPKBG(KFIL10,KFIL12,IA,NDX,LOC,IPOS,NVALUE,NBIT,
     1                  IER,*)
C
C        APRIL 1993   GLAHN   TDL   MICROVAX
C
C        PURPOSE
C            UNPACKS NBIT BITS INTO THE WORD NVALUE FROM ARRAY
C            IA(NDX) STARTING IN WORD LOC, BIT IPOS.  THE WORD
C            POINTER LOC AND BIT POSITION POINTER IPOS ARE UPDATED
C            AS NECESSARY.  IF NBIT EQ 0, NVALUE IS RETURNED EQ 0,
C            AND THE POINTERS ARE NOT MOVED.
C
C        DATA SET USE
C           KFIL10 - UNIT NUMBER FOR CURRENT CONSOLE. (OUTPUT)
C           KFIL12 - UNIT NUMBER FOR OUTPUT (PRINT) FILE. (OUTPUT)
C
C        VARIABLES
C              KFIL10 = UNIT NUMBER FOR CURRENT CONSOLE.  (INPUT)
C              KFIL12 = UNIT NUMBER FOR OUTPUT (PRINT) FILE.  (INPUT)
C               IA(J) = ARRAY TO UNPACK FROM (J=1,NDX).  (INPUT)
C                 NDX = DIMENSION OF IA( ).  (INPUT)
C                 LOC = WORD IN IA( ) TO START UNPACKING.  UPDATED
C                       AS NECESSARY AFTER UNPACKING IS COMPLETED.
C                       (INPUT-OUTPUT)
C                IPOS = BIT POSITION (COUNTING LEFTMOST BIT IN WORD
C                       AS 1) TO START UNPACKING.  MUST BE GE 1 AND
C                       LE 32.  UPDATED AS NECESSARY
C                       AFTER PACKING IS COMPLETED.  (INPUT-OUTPUT)
C              NVALUE = THE RIGHTMOST NBIT BITS IN NVALUE WILL
C                       BE FILLED FROM IA(LOC).  IA(LOC+1) IS USED IF
C                       NECESSARY.  RETURNED AS ZERO IF IER NE 0.
C                       (OUTPUT)
C                NBIT = SEE NVALUE.  MUST BE GE 0 AND LE 32.  (INPUT)
C                 IER = STATUS RETURN:
C                       0 = GOOD RETURN.
C                       1 = LOC NOT IN RANGE 1 TO NDX.
C                       2 = IPOS NOT IN RANGE 1 TO 32.
C                       3 = NBIT NOT IN RANGE 0 TO 32.
C                     * = ALTERNATE RETURN WHEN IER NE 0.
C
C        NON SYSTEM SUBROUTINES CALLED
C            NONE (VAX FORTRAN 77 EXTENSIONS ISHFT AND IOR ARE USED)
C
      DIMENSION IA(NDX)
C
C        CHECK CORRECTNESS OF INPUT AND SET STATUS RETURN.
C
      IER=0
      NVALUE=0
      IF(NBIT.EQ.0)GO TO 150
C        IF IER NE 0, NVALUE IS RETURNED EQ 0.
      IF(LOC.GT.NDX)IER=1
      IF(IPOS.LE.0.OR.IPOS.GT.32)IER=2
      IF(NBIT.LT.0.OR.NBIT.GT.32)IER=3
      IF(IER.NE.0)RETURN 1
```

```
C        SHIFT WORD IA(LOC) TO LEFT TO ELIMINATE BITS TO LEFT OF THOSE
C        WANTED, THEN BACK TO THE RIGHT TO THE PORTION OF THE WORD
C        WANTED.
C
      NVALUE=ISHFT(ISHFT(IA(LOC),IPOS-1),NBIT-32)
C
C        UPDATE IPOS AND LOC AS NEEDED.
C
      IPOS=IPOS+NBIT
      IF(IPOS.LE.32)GO TO 150
      LOC=LOC+1
      IPOS=IPOS-32
      IF(IPOS.EQ.1)GO TO 150
C
C        FINISH UNPACKING.
C
      IF(LOC.LE.NDX)GO TO 130
      IER=1
      NVALUE=0
      RETURN 1
C
 130  NVALUE=IOR(NVALUE,ISHFT(IA(LOC),IPOS-33))
C
 150  CONTINUE
      RETURN
      END
```

APPENDIX III

This appendix summarizes the changes that would have to be made in the Manual on Codes (WMO 1988) to implement the four options recommended in this office note.

Flag/Code Table 8 - A 1 in bit 4 would be used to indicate the alternate reversal of scanning. Bits 1, 2, and 3 could keep their same meanings.

Code Table 11-Flag - A 1 in bit 9 (which is really packed as bit 5 in octet 14 of Section 4) would be used to indicate that second-order spatial differences are packed.

Section 4, Gridpoint Data, Second-Order Packing - No change through octet 21. Then the order of data would be:

Octet 22: Contains the width of the group widths (the number of bits needed to pack the largest of the widths).

Octet 23 and following: P1 (octets 17-18) widths of octet 22 size.

Next 8 bits: Contains the number of bits needed (say, MBIT) to provide the group size information, followed by P1 group sizes, each MBIT's in size.

The secondary bit map would be deleted. The above group information could be padded with zeros to a whole octet.

As before, the packed gridpoint values (with the overall and group minima removed) follow. That is, P1 first-order values padded with zeros to a whole octet, then P2 second-order values.

Section 4, Gridpoint Data, Second-Order Packing, Second-Order Spatial Differences - No change from the above down to the packing of the actual values. Then the order would be:

After the P1 group sizes and the pad, use 5 bits to hold the number of bits (say, IBIT) needed to hold the first value in the field (this is the first value, with both the overall and first group minimum subtracted; this value will be positive). Then use IBIT's to pack this first value.

Next, use 5 bits to hold the number of bits (say, JBIT) needed to hold the absolute value of the first first-order spatial difference. Use the next bit to hold the sign (1 for negative) of this first first-order difference, then pack the absolute value of the first first-order difference in JBIT's.

Then, use 5 bits to hold the number of bits (say, NBIT) needed to hold the absolute value of the minimum second-order spatial difference. Use the next bit to hold the sign of this minimum, then pack the absolute value of this minimum in NBIT's.

Finally, the second-order spatial differences are packed in the same method as before. That is, P1 first-order values padded with zeros to a whole octet, then P2-2 second-order spatial differences. Note that there will be two less second-order spatial differences than original values. As usual,

any group in which all values are the same will not require these values to be packed, only the minimum.

It is noted that when second-order spatial differences are used, the groups are determined relative to those differences, rather than relative to the original values. In this case, the overall minimum (reference) value in octets 7-10 serves no useful purpose, but is retained here for consistency and simplicity. In packing, it can be subtracted out before it is determined whether or not to use second-order spatial differences.