UNPK_GRIB2

UNPACKS DATA FROM GRIB2 FORMAT

Bryon Lawrence
April 5, 2001
Bob Glahn
David Rudack
March 15, 2002

PURPOSE: To unpack a gridded data field and its associated defining data from a GRIB2 message using the algorithm put forth in version two of the World Meteorological Organization's (WMO) standard for the exchange of General Regularly-distributed Information in Binary form (GRIB2). In addition to decoding and returning a gridded data field, this GRIB2 decoder also returns information that identifies and defines the data field. Such information includes the time of generation of the gridded product, the source of the gridded product, the type of map projection the gridded product uses, what the data in the gridded product represent, and which packing method was used to compress the data in the gridded product.

Depending on the type of data contained within the data field, the unpacked data field is returned to the user either in a floating point or integer array. The additional data defining and identifying the gridded product are returned to the user through eight integer "section" arrays. Each of these arrays corresponds to one of the sections (**Sections 0, 1, 2, 3, 4, 5, 6,** and **7**) that form the structure of a GRIB2 message. Note that no information is returned from this unpacker concerning **Section 8** (the **End Section**); this GRIB2 section is only used internally by the unpacker. Also note that **Section 2**, the **Local Use Section**, is optional and may not be present in the GRIB2 message.

The simple, complex, and complex with second order spatial differencing data packing schemes are recognized by this routine. If the simple packing method was used to create the GRIB2 message, then the unpacked data field may contain **primary missing values**. If the complex or complex with second order spatial differences packing method was used to compress the data, then the unpacked data field returned may contain both **primary** and **secondary missing values**.

As a carry over from GRIB version 1, GRIB2 continues to allow the use of a bit-map (or bit-mask) to indicate the positions of **primary missing values** in a data field.

However, the MDL GRIB2 decoder only allows the use of a bit-map while unpacking data using the simple or complex methods. The use of a bit-map with complex with second order spatial differences is not supported. The complex packing methods now have an alternative, and usually more space efficient, way of dealing with missing values that eliminates the need for a bit-map. However, it is still necessary to use a bit-map to locate the positions of missing values in a data field that was packed using the simple packing method.

When unpacking a data field that was packed using the simple packing method, and there is a bit-map accompanying the data field, the user is given the option of having the data field returned with the primary missing values embedded in it or without the primary missing values. This is accomplished by manipulating the value of the "ICLEAN" calling argument (see below).

More than one data grid may be contained within a GRIB2 message. This routine provides the functionality needed to unpack multiple data grids from a single GRIB2 message. This is accomplished through repetitive calls to this routine with the "NEW" calling argument (see below) properly set while testing the value of the "IENDPK" calling argument (again, see below) after each call to this routine. According to WMO GRIB2 regulations, **Sections 2** through **7**, **3** through **7**, or **4** through **7** may be repeated for each data grid packed into the GRIB2 message.

For a complete description of the GRIB2 format, templates, and code tables, the user is referred to the WMO document **FM 92-XII GRIB**.

CALL AND EXPLANATION OF FORMAL PARAMETERS:

```
CALL UNPK_GRIB2(KFILDO,A,IA,ND2X3,IDAT,NIDAT,RDAT,
1           NRDAT,IS0,NS0,IS1,NS1,IS2,NS2,IS3,NS3,
2           IS4,NS4,IS5,NS5,IS6,NS6,IS7,NS7,IB,
3           IBITMAP,IPACK,ND5,XMISSP,XMISSS,NEW,
4           ICLEAN,L3264B,IENDPK,JER,NDJER,KJER)
```

KFILDO - Unit number of the output diagnostic (print) file. All lines of source that create diagnostic output in the unpacker routine are "commented out" with a "D" in column 1 of the source code. If the user desires that diagnostic information be generated when this unpacker is executed, then the option specific to the Fortran compiler being used to compile the unpacker library that allows the compilation of debug lines as source code must be used. When diagnostic information is desired, the user must make sure that the file represented by this number has been opened **prior** to calling the “**unpk_grib2**" routine. (INPUT)

A(L) - The unpacked gridpoint data are returned to the caller in this array when the original packed data field consisted of floating point values (L=1,ND2X3). If the original packed data field consisted of floating point values, then element 21 of the IS5( ) array will contain a value of “0". If the original packed data field consisted of integer values, then the unpacked grid point data will be returned to the caller of this routine in the IA( ) array. (OUTPUT)

IA(L) - The unpacked grid point data are returned to the caller of this routine in this array when the original packed data field consisted of integer values (L=1,ND2X3). If the original packed data field consisted of integer values, then element 21 of the IS5( ) array will contain a value of “1". If the original packed data field consisted of floating point values, then the unpacked grid point data will be returned to the caller of this routine in the A( ) array. (OUTPUT)

ND2X3 - The dimension of A( ), IA( ), and IB( ). It should be at least the same size as the number of grid points in the data field. (OUTPUT)

IDAT(L) - Contains the integer local use data (if any) that were unpacked from **Section 2**, the **Local Use Section**, of the GRIB2 message (L=1, NIDAT). See the special documentation below describing the format of the local use data returned by the MDL GRIB2 decoder. (OUTPUT)

NIDAT - The number of elements in the IDAT( ) array. Must be made large enough to contain any integer value local use data contained within the GRIB2 message. (INPUT)

RDAT(L) - Contains the floating point local use data (if any) that were unpacked from **Section 2**, the **Local Use Section**, of the GRIB2 message (L=1,NRDAT). See the special documentation below detailing the format of the local use data returned by the MDL GRIB2 decoder. (OUTPUT)

NRDAT - The number of elements in the RDAT( ) array. Must be large enough to contain any floating point local use data unpacked from the GRIB2 message. (INPUT)

IS0(L) - Contains the unpacked values for **Section 0**, the **Indicator Section**, of the GRIB2 message being decoded (L=1,NS0). See the GRIB2 section outline below for an overview of the contents of **Section 0**. (OUTPUT)

NS0 - The dimension of IS0( ). NS0=16 is sufficient. (INPUT)

IS1(L) - Contains the unpacked data values for **Section 1**, the **Identification Section** (1,NS1). See the GRIB2 section outline below for an overview of the contents of **Section 1**. (OUTPUT)

NS1 - The dimension of IS1( ). NS1=21 is sufficient. (INPUT)

IS2(L) - Contains the length, section number, and total number of local use data groups that were unpacked from **Section 2**, the **Local Use Section** (L=1,NS2). If no local use section exists, then the length of **Section 2** will be reported as “0." The first elements of the IDAT( ) and RDAT( ) arrays will also have values of “0". See the special documentation below concerning **Section 2** and the format of local use data returned from this routine. (OUTPUT)

NS2 - Dimension of IS2( ). It must be a large enough dimension to contain any data the that may be unpacked from the local use section. (INPUT)

IS3(L)- Contains the unpacked data values for **Section 3**, the **Grid Definition Section**, of the GRIB2 message being decoded (L=1,NS3). Section 3 defines the type of map projection that the data field uses. See the GRIB2 section outline below for an overview of the contents of **Section 3**. (OUTPUT)

NS3 - The number of elements in the IS3( ) array. Since the grid definition templates are of variable size, the value of this parameter depends upon what type of map the data field being unpacked is projected on. NS3=96 is sufficient for all templates except template 3.120, the Azimuth-Range Projection, which is commonly used for radar images. In the case where template 3.120 is being used, NS3=1600 should be sufficient. (INPUT)

IS4(L)- Contains the unpacked data values for **Section 4**, the **Product Definition Section**, of the GRIB2 message being decoded (L=1,NS4). Section 4 defines what the data field being unpacked represents, i.e. do the data represent a map of 1-hour rainfall totals or do they represent the height contours on an Aviation Model 500-mb height forecast grid. See the GRIB2 section outline below for an overview of the contents of GRIB2 and **Section 4**. (OUTPUT)

NS4 - The number of elements in the IS4( ) array. Since the product definition templates are of variable size, the value of this parameter depends on what type of product is represented by the data field being unpacked from the GRIB2 message. For this GRIB2 decoder, NS4=60 should be sufficient for all of the supported Section 4 product definition templates, with the possible exception of **template 4.30**, the **Satellite Product**, which could require more array space depending on the number of contributing bands in the satellite image. (INPUT)

IS5(L) - Contains the unpacked data for **Section 5**, the **Data Representation Section** (L=1,NS5). **Section 5** indicates which packing method was used to pack the gridded data field into the GRIB2 message. See the GRIB2 section outline below for an overview of the contents of **Section 5**. (OUTPUT)

NS5 - The dimension of the IS5( ) array. NS5=49 is sufficient for all of the packing methods recognized by this decoder. (INPUT)

IS6(L) - Contains the unpacked data for **Section 6**, the **Bit-map Section** (L=1,NS6). These data consist of information specifying the length of **Section 6**, the section number, and a bit-map indicator which indicates whether or not a bit-map is present in the GRIB2 message. Note that the actual bit-map, if it exists, is returned to the caller in the IB( )array. See the GRIB2 section outline below for an overview of the contents of **Section 6**. (OUTPUT)

NS6 - Size of IS6( ). NS6=6 is sufficiently large for all products. (INPUT)

IS7(L) - Contains the unpacked data for **Section 7**, the **Data Section** (L=1,NS7). See the GRIB2 section outline below for an overview of the contents of **Section 7**. Note that the actual unpacked gridded data are not returned via this array. Rather they are returned in the A( ) or IA( ) arrays (see above) depending on the type of the gridded data (as indicated by octet 21 of **Section 5**). (OUTPUT)

NS7 - Dimension of IS7( ). A value of at least “8" is required for this parameter. (INPUT)

IB(L) - Contains a bit-map indicating the locations of **primary missing values** in the unpacked data grid. A bit-map will be returned to the user only when a bit-map was packed into the GRIB2 message and the user requests it (see ICLEAN above). (OUTPUT)

IBITMAP - Indicates whether or not a bit-map is being returned from this routine in the IB( ) array. A value of “0" means that a bit-map is not being returned. A value of “1" means that a bit-map is being returned. (OUTPUT)

IPACK(L)- The GRIB2 message to be unpacked is supplied by the caller of this routine in this array (L=1,ND5). (INPUT)

ND5 - Dimension of IPACK( ). Must be dimensioned large enough to contain the entire packed product. (INPUT)

XMISSP- The floating point representation of the **primary missing value**. This value is set by the unpacker when either the complex or complex with second order spatial differences packing methods was used to create the GRIB2 message and element 23 of the IS5( ) array indicates that there are primary missing values. A value will not be returned in this parameter if the simple packing method was used to create the GRIB2 message. **If the simple packing method was used, and the user wants the unpacked data field returned with the missing values embedded within it, then the value to use to indicate a missing datum must be passed into this routine by this parameter.** (INPUT/OUTPUT)

XMISSS - The floating point representation of the **secondary missing value**. This value is set by the decoder when either the complex or complex with second order spatial differences packing methods was used to create the GRIB2 message and element 23 of the IS5( ) array indicates that there are primary and secondary missing values in the data field. A value will not be returned in this parameter if the simple packing method was used to create the GRIB2 message. **With the complex packing method, second-**

ary missing values are only allowed if there can be primary missing values. Secondary missing values are not supported with the simple packing method or with the complex and second order spatial differences packing method. (OUTPUT)

NEW - Indicates whether or not this is the first data grid to be unpacked from a GRIB2 message. A value of "1" indicates that this is the first grid to be unpacked. A value of "0" indicates that this is not the first grid to be unpacked. When unpacking GRIB2 messages that contain only one packed data grid, this parameter must always be "1". When unpacking a GRIB2 message that contains more than one packed data grid, then this parameter must be "1" on the first call to the packer and then "0" on all subsequent calls to the unpacker. (INPUT)

ICLEAN - A flag that applies only if the simple or the complex packing method was used to create the GRIB2 message **and there was a bit-map packed into this message. It does not apply to complex packing with second order spatial differences.** A value of "1" in this calling argument means that the user wants the unpacked data field to be returned without any missing values in it. A value of "0" means that the user wants the unpacked data field to be returned with the missing values embedded in it. (INPUT)

L3264B - Integer word length in bits of the machine being used (either 32 or 64). (INPUT)

IENDPK - Parameter indicating whether or not there are additional data grids to be unpacked within the GRIB2 message. A value of "1" means that the unpacking of the GRIB2 message is complete; there are no more data grids to unpack from the GRIB2 message. A value of "0" means that the there are additional data grids to be unpacked from the GRIB2 message. It will be necessary to call the **"unpk_grib2"** routine **at least** once more to completely unpack the GRIB2 message. (OUTPUT)

JER(L,M)- Contains any diagnostic or error codes along with their severity levels generated in this routine (L=1,NDJER) (M=1,2). This error-handling scheme was developed to preserve all diagnostic information generated during the execution of this routine. Since some error codes are non-fatal and offer information that is of diagnostic value, it is possible that a run of this GRIB2 decoder may generate several diagnostic codes. This array provides the user with a way of deducing an error “trace back.” This error handling scheme works as follows:

The rows in the JER array represent individual error occurrences. The first column in the JER array represents the error code; the second column represents the severity of the error code.

There are three severity levels that can be assigned to an error code:

0 = Not a Problem
1 = Warning
2 = Fatal

An error with a severity level of “Warning” does not warrant the termination of the unpacker. An error with a severity level of “Fatal” results in the termination of the unpacker even if the GRIB2 message has not been completely unpacked.

Each time the unpacker starts unpacking a new section of the GRIB2 message, it places a three digit section code representing the section being decoded followed by a severity level of “0" into the first and second columns, respectively, of the next available row of the JER array. Section codes are 0 (Section 0), 100 (Section 1), 200 (Section 2), 300 (Section 3), 400 (Section 4), 500 (Section 5), 600 (Section 6), 700 (Section 7), and 800 (Section 8).

When an error is encountered while “degribbing” a message, the routine detecting the error will place the error code followed by its severity level into the first and second columns, respectively, of the next available row of the JER array.

For example, suppose a call to “unpkbg” failed while unpacking Section 7 of a GRIB2 message because the “NBIT” calling argument did not have a value inclusively contained in the range of 0 to 32. The contents of the JER array upon being returned to the caller of the “unpk_grib2" subroutine would appear as follows:

Contents of JER	Diagnostic/Error Code	Severity
row 1	0	0
row 2	100	0
row 3	200	0
row 4	300	0
row 5	400	0
row 6	500	0
row 7	600	0
row 8	700	0
row 9	8	2

This tells the user that all sections **up to** Section 7 were successfully unpacked. Note that the diagnostic/error code 0 corresponds to Section 0; 100 corresponds to Section 1; 200 corresponds to Section 2; 300 corresponds to Section 3; 400 corresponds to Section 4; 500 corresponds to Section 5; 600 corresponds to Section 6; and 700 corresponds to Section 7. Also note that since each of these section codes is followed by a severity level of 0, it means that the unpacking of the GRIB2 message has been successful **UP TO THAT SECTION.** The error code of "8" in row 9 is the error code generated by routine **"unpkbg"** indicating the invalid value of the "NBIT" calling argument. The "2" in the severity column indicates that the error is fatal and that the decoding of the GRIB2 message is being halted with return to the user.

The advantage to using this error handling scheme is that the caller of the **"unpk_grib2"** routine can isolate where the problem occurred (in this example, Section 7). This problem would be very difficult to find if the user was given a single error code upon return from the **"unpk_grib2"** routine especially since the **"unpkbg"** utility is called throughout the entire decoder. This error handling scheme was created to give the user some type of error handling/traceback capability in lieu of the unpacker actually printing out diagnostic messages. However, if the user desires diagnostic output, see the notes corresponding to the "KFILDO" calling argument above. (OUTPUT)

NDJER - The number of rows in JER( ). It is recommended that this be set to at least “15". If this value is not set large enough and the JER array fills up, the last row of the JER array will be overwritten with an error code of “999" with a fatal severity level of “2". This will result in the loss of at least two diagnostic codes and their corresponding severity levels. The decoding of the GRIB2 message will be halted. (INPUT)

KJER - The number of error/diagnostic messages contained within the JER array. Useful for testing for errors when program control is returned from the **“unpk_grib2"** routine back to the calling routine. (OUTPUT)

OUTPUT:

Diagnostic messages will be written to Unit No. “KFILDO” when pk_grib2 has been compiled using the compile options as outlined above in the description of the “KFILDO” calling argument above.

RESTRICTIONS:

Because of using floating point computations for unpacking, exact values of packed integers may not be preserved for very large numbers (2**24-1 seems to work ok, but ≥ 2**25-1 does not).

NONSYSTEM ROUTINES USED:

See the user associated library.

LANGUAGE: FORTRAN 90

GRIB2 FORMAT:

Nine sections are defined for GRIB2. Sections in ( ) are optional.

Section	Section Name	Section Contents
0	Indicator Section	“GRIB”, GRIB edition #, message length
1	Identification Section	Characteristics of all the processed data
(2)	(Local Use Section)	Additional items for local use
	Grid Definition	

3	Section	Geometry of values
4	Product Definition Section	Description of following processed data
5	Data Representation	How the processed data are packed
6	Bit-map Section	Indicator of value being present/absent
7	Binary Data Section	Binary data values
8	End Section	“7777"

The contents of each section of the GRIB2 message, as well as number of octets (bytes) required to store each element in the section are detailed in the WMO document **FM 92-XII GRIB**. Arrays named IS0 - IS7 are used to pass the required input/output values into and out of the packer. Each of these arrays corresponds to a section in the GRIB2 message, e.g. array IS0 corresponds to **Section 0**. The element number in each of these “IS” arrays corresponds to the beginning octet number where the data value is stored in the section. So, for example, a value that is stored beginning in octet 5 of Section 5 would be placed into element 5 of the IS5 array.

Section 0:

IS0(1) = GRIB name, stored in bytes 1-4
IS0(7) = Discipline - master table number, stored in byte 7
IS0(8) = GRIB edition number, stored in byte 8
IS0(9) = Total length of the GRIB message, stored in bytes 9-16

Section 1:

IS1(1) = Length of section, stored in bytes 1-4
IS1(5) = Number of section, stored in byte 5
IS1(6) = ID of originating/generating center, stored in bytes 6-7
IS1(8) = ID of originating/generating sub-center, stored in bytes 8-9
IS1(10) = GRIB Master tables version number (0), stored in byte 10
IS1(11) = GRIB Local tables version number (0), stored in byte 11
IS1(12) = Significance of reference time stored in byte 12
IS1(13) = Year (4 digits), stored in bytes 13-14
IS1(15) = Month, stored in byte 15
IS1(16) = Day, stored in byte 16
IS1(17) = Hour, stored in byte 17
IS1(18) = Minute, stored in byte 18
IS1(19) = Second, stored in byte 19

IS1(20) = Production status of processed data in message, stored in byte 20
IS1(21) = Type of processed data in message, stored in byte 21

Section 2:

IS2(1) = Length of section, stored in bytes 1-4
IS2(5) = Number of section, stored in byte 5
IS2(6) - IS2(nn) = Local use, stored in bytes 6-nn

Section 3:

IS3(1) = Length of section, stored in bytes 1-4
IS3(5) = Number of section, stored in byte 5
IS3(6) = Source of grid definition, stored in byte 6
IS3(7) = Number of data points, stored in bytes 7-10
IS3(11) = Number of octets for optimal list of numbers defining number of points, stored in byte 11
IS3(12) = Interpretation of list of numbers defining number of points, stored in byte 12
IS3(13) = Grid Definition Template Number, stored in bytes 13-14
IS3(15) - IS3(nn) = Grid Definition Template, stored in bytes 15-nn

Section 4:

IS4(1) = Length of section, stored in bytes 1-4
IS4(5) = Number of section, stored in byte 5
IS4(6) = Number of coordinates values after Template, stored in bytes 6-7
IS4(8) = Product Definition Template Number, stored in bytes 8-9
IS4(10) - IS4(nn) = Product Definition Template, stored in bytes 10-nn

Section 5:

IS5(1) = Length of section, stored in bytes 1-4
IS5(5) = Number of section, stored in byte 5
IS5(6) = Number of data points where one or more values are specified in Section 7 when a bit map is present, total number of data points when a bit map is absent, stored in bytes 6-9
IS5(10) = Data Representation Template Number, stored in bytes 10-11
IS5(12) - IS5(nn) = Data Representation Template, stored in bytes 12-nn

Section 6:

IS6(1) = Length of section, stored in bytes 1-4
IS6(5) = Number of section, stored in byte 5
IS6(6) - IS6(nn) = Bit map stored in bytes 6-nn

Section 7:

IS7(1) = Length of section, stored in bytes 1-4
IS7(5) = Number of section, stored in byte 5

IS7(6) - IS7(nn) = Binary Data Values, stored in bytes 6-nn

Section 8:

IS8(1) = “7777", stored in bytes 1-4

Local Use Data (Section 2)

GRIB2 provides the capability to preserve and pass along information about the gridded data field that the GRIB2 format does not provide specific templates or code tables for. **Section 2** is provided to contain these local use data. GRIB2 does not specify any restrictions on the format of the data in Section 2, which gives much flexibility in determining how the data are stored. The local use data processing scheme described below is the one employed by Version 1 of the MDL GRIB2 encoder/decoder software.

Because the format of the data in **Section 2** is specific to the originating source of the GRIB2 message, this unpacking routine will skip over any local use data in **Section 2** if it detects that it was not packed using Version 1 of the MDL GRIB2 encoder. MDL GRIB2 decoding software examines octet 6 of Section 2 to determine if the local use data was packed by the MDL GRIB2 software. If the data format is not recognized, this routine will log an error code of “208" and a severity level of “1" in the JER(,) error handling array. The severity level of “1" implies that this is not a fatal error, but it does alert the user that there were data in the **Local Use Section** that were not unpacked.

The MDL GRIB2 encoder/decoder employ a flexible format for storing local use data that allows for the storage of both integer and floating point groups of values. In addition to this, the local use data are packed using the **simple packing method**. Since the local use data are broken down into individual groups, the user is given the power to specify the decimal scaling factor for each group, thus providing the ability to optimize the compression of each of the local use data groups based upon the type and precision of the data that they contain.

Upon retrieval of the local use data during the unpacking of a GRIB2 message that was originally packed using MDL GRIB2 software, the integer and floating point local use data will be stored, respectively, in the IDAT( ) and RDAT( ) array calling arguments (see descriptions of the calling arguments above). If there are no local use data (**Section 2 is optional**), then the first element of the IDAT( ) and RDAT( ) arrays will contain a value of “0" and the length of **Section 2** will be indicated by a value of “0" in IS2(1). If there are local use data, element 7 of the IS2( ) array will contain the total number of groups of local use data with the actual data groups being returned in the IDAT( ) and RDAT( ) arrays using the following format:

Array Element Number	Description of Content
1	Number of values in the first group of data (N1).
2	The decimal scale factor that was used in packing the first group of local use data.
3 to (N1+2)	First group of local use data values.
N1+3	Number of values in the second group of local use data (N2).
N1+4	The decimal scale factor that was used in packing the second group of local use data.
(N1+5) to (N1+N2+4)	Second group of local use data values.
(K-1)*2+1+N1+N2+...+N(k-1)	Number of values in the Kth group of data (Nk)
(K-1)*2+2+N1+N2+...+N(k-1)	The decimal scale factor that was used in packing the Kth group of data.
(K-1)*2+3+N1+N2+...+N(k-1) to (K-1)*2+N1+N2+...+N(k-1)+Nk)	The Kth group of local use data values.
(K-1)*2+1+N1+N2+...+Nk)	“0" A value of “0" where the size of the group goes means that there is no more local use data supplied in this array.

Processing GRIB2 messages on Systems with Different Memory Architectures

A few extra steps must be taken when using this software on a system that uses a “little-endian” memory architecture, where the low-order byte of a word represents the word’s starting address. The order of bytes in the message is at least implied to be high order first and specifically the WMO documentation reads concerning floating point, "The numbers are stored with the high order octet first. The sign bit will be the first bit of the first octet. The low order bit of the mantissa will be the last (eighth) bit of the fourth octet. This floating point representation has been chosen because it is in common use in modern computer hardware. Some computers use this representation with the order of the octets reversed. They will have to convert the representation, either by reversing the octets or by computing the floating point value directly..." PK_GRIB2 uses this "big endian" representation.

The following applies to applications using this GRIB2 decoder when decoding a GRIB2 message that is in "big-endian" format.

Before commencing to unpack a GRIB2 message on a "little-endian" system, it is essential that the bytes in the packed GRIB2 message be "swapped" to conform to "little-endian" standards. This should be done just before calling the "unpk_grib2" routine. A routine, named "unpk_swap", is provided in the unpacker library to perform this byte swapping on the GRIB2 message. It takes as arguments the array containing the packed message and the number of elements in that array. The byte swapping is performed in-situ within this array.

The "unpk_swap" routine is written in the C language for greater efficiency. Exactly how this routine is linked into the user's executable depends upon the linker that is being used and the language that the main routine is written in. When using a Fortran main routine, it is recommended that the user see the "man" pages supplied with the compiler/linker being used to build the unpacker library for information on how to call a "C" routine from a Fortran main routine.

If the user is uncertain of the type of memory architecture of the system the GRIB2 message is being unpacked on, then function "unpk_endian" (also supplied in the unpacker library) should be called. Function "unpk_endian" will return a value of "TRUE" on a "big-endian" system and a value of "FALSE" on a "little-endian" system. The following is a portion of a Fortran driver demonstrating how to use the "unpk_swap" and "unpk_endian" routines:

```
      PROGRAM UNPACKER
C
      LOGICAL BIG
      ...
C
      BIG=UNPK_ENDIAN()
C
      IF(.NOT.BIG)THEN
         CALL UNPK_SWAP(IPACK,ND5)
      ENDIF
C

      CALL UNPK_GRIB2(KFILDO,AIN,IAIN,NX,NY,IDAT,NIDAT,RDAT,NRDAT,
     1             IS0,NS0,IS1,NS1,IS3,NS3,IS4,NS4,IS5,NS5,IS6,NS6,
     2             IS7,NS7,IB,IBITMAP,IPACK,ND5,MISSP,XMISSP,MISSS,
     3              XMISSS,NEW,MINPK,ICLEAN,L3264B,JER,NDJER,KJER)
C
      IF(JER(KJER,2).EQ.2)THEN
         WRITE(*,15) JER(KJER,1)
 15      FORMAT(//,1X,'FATAL ERROR IN PK_GRIB2 ERROR CODE ',I5)
         STOP
      ENDIF
C
      ...
```

Error Codes

The following is a list of all of the possible diagnostic and error return codes that can be returned by this routine. An attempt has been made to give the numeric codes a meaningful appearance that will aid the user of this GRIB2 decoder in identifying which part of the GRIB2 software is generating the error condition. For example, errors encountered while unpacking Section 3 of the GRIB2 message will generally have a value ranging from 301 to 399 while those encountered while unpacking Section 4 of the GRIB2 message will generally have a value ranging from 401 to 499. Not all error codes represent “fatal” circumstances that require the termination of the unpk_grib2 routine. Along with the actual value of an error code, severity information is also returned to the user indicating whether the error represents a “fatal” condition or is provided just for diagnostic purposes.

0 = A good return value **(all routines)**
6 = LOCN is not in the range 1 to ND5 **(unpkbg.f, unpkoo.f, unpkpo.f, unpkps.f)**
7 = IPOS is not in range 1 to L3264B **(unpkbg.f, unpkcmbm.f, unpklxbm.f, unpkoo.f, unpkpo.f, unpkps.f)**
8 = NBIT not in range 0 to 32 **(unpkbg.f, unpkcmbm.f, unpklxbm.f, unpkoo.f, unpkpo.f, unpkps.f)**
9 = NDX is not large enough to furnish the bits necessary to accommodate NXY values starting at the values LOCN and IPOS **(unpklxbm.f)**
18 = Unrecognized missing value code in IMISSING **(unpkcmbm.f)**
100 = No Section “1" on a new product **(unpk_sect1.f)**
102 = IS1( ) has not been dimensioned large enough to contain the entire template **(unpk_sect1.f)**
199 = Unexpected end of message **(unpk_sect1.f)**
202 = IS2( ) has not been dimensioned large enough to contain the entire template **(unpk_sect2.f)**

204 = The number of local use data groups specified in octet 6 of Section 2 is “0" **(unpk_sect2.f)**
206 = Invalid data type indicator value in Section 2 **(unpk_sect2.f)**
208 = The data in Section 2 has an unrecognized format. It has been skipped and not unpacked (non-fatal error return) **(unpk_sect2.f)**
299 = Unexpected end of message **(unpk_sect2.f)**
300 = IS3(5) does not indicate Section 3 **(unpk_sect3.f, unpk_grib2.f)**
302 = IS3( ) has not been dimensioned large enough to contain the entire template **(unpk_sect3.f, unpk_azimuth.f, unpk_cylinder.f, unpk_equator.f, unpk_lambert.f, unpk_mercator, unpk_orthographic.f, unpk_polster.f)**
303 = Unsupported grid template indicated by IS3(13) **(unpk_azimuth.f, unpk_cylinder.f, unpk_equator.f, unpk_lambert.f, unpk_mercator.f, unpk_orthographic.f, unpk_polster.f, unpk_sect3.f)**
307 = Unrecognized or unsupported shape of Earth code in IS3(15) **(unearth.f, unpk_sect3.f, unpk_cylinder.f, unpk_equator.f, unpk_lambert.f, unpk_mercator.f, unpk_orthographic.f, unpk_polster.f**)
399 = Unexpected end of message **(unpk_sect3.f)**
401 = IS4(5) does not indicate Section 4 **(unpk_sect4.f)**
402 = IS4( ) has not been dimensioned large enough to contain the template **(unpk_sect4.f, unpk_temp40.f, unpk_temp41.f, unpk_temp430.f, unpk_temp48.f, unpk_temp420.f**)
403 = Not the correct template or unsupported template **(unpk_sect4.f, unpk_temp41.f, unpk_temp430.f, unpk_temp48.f, unpk_temp420.f)**
499 = Unexpected end of message **(unpk_sect4.f)**
501 = IS5(5) does not indicate Section 5 **(unpk_sect5.f)**
502 = IS5( ) has not been dimensioned large enough to contain the template **(unpk_sect5.f)**
508 = Unrecognized or unsupported type of packing in IS5(10) **(unpk_sect5.f)**
509 = Unrecognized type of data in IS5(21) **(unpk_sect5.f)**
599 = Unexpected end of message **(unpk_sect5.f)**
601 = IS5(5) does not indicate Section 6 **(unpk_sect6.f)**
602 = IS6( ) has not been dimensioned large enough to contain the entire template **(unpk_sect6.f)**
605 = Dimension ND2X3 not large enough for IB(NX*NY) **(unpk_sect6.f)**
608 = Bit-map indicator in IS6(6) not in correct range **(unpk_sect6.f)**
699 = Unexpected end of message **(unpk_sect6.f)**
701 = IS7(5) does not indicate Section 7 **(unpk_sect7.f)**

702 = IS7( ) has not been dimensioned large enough to contain the entire template **(unpk_sect7.f)**
705 = ND2X3 is too small of a dimension to allow for proper processing of the grid **(unpk_cmplx.f, unpksecdif.f)**
708 = Invalid unpacking option **(unpk_refer.f)**
709 = Unsupported order of differencing **(unpk_cmplx.f)**
799 = Unexpected end of message **(unpk_sect7.f)**
999 = The JER( ) array is full **(unpk_trace.f)**
1002 = IS0( ) has not been dimensioned large enough to contain the entire template **(unpk_sect0.f)**
1010 = The beginning of GRIB message was not found **(unpk_sect0.f)**
1011 = Message must be GRIB2 version 2 **(unpk_sect0.f)**

Version Control:

This Version 1.1 has been modified from Version 1.0 only 1) to add more diagnostic messages, 2) to improve readability and comments, and 3) to correct any deficiencies found through more rigorous testing. GRIB offers many options, only a portion of which are supported by PK_GRIB; even so, the number of combinations is large and some errors could still remain.