平成17年4月28日
気象庁予報部

配信資料に関する技術情報(気象編)第196号

～毎時大気解析GPVの提供について～

1時間毎に風と気温の解析を行い、その結果を「毎時大気解析」として平成17年7月1日から提供開始します。

1. 新たに提供を開始するデータ

毎時大気解析による風と気温の解析値ファイルをFTP方式で提供します。データ概要は添付資料1を、データの特徴は添付資料2を参照してください。

なお、気温データの配信は平成18年3月からで、それまでは気温のデータ部はありません。また、毎時大気解析は平成18年3月以降に計画している次期メソ数値予報モデルでの配信プロダクトと同じ格子系(5km)で提供します。

2. 提供開始日時

平成17年7月1日(金)00UTCから配信を開始します。なお、提供開始に先立ち6月23日(木)から試験配信を開始します。

毎正時後30分以内に気象業務支援センターへの配信を完了します。なお、当庁のシステムに障害が発生し翌正時までに配信を完了できない場合は、当該時刻のファイルの配信を中止します。

3. 毎時大気解析GPVのファイル名

Z__C_RJTD_*yyyyMMddhhmmss*_QMA_GPV_Rjp_ANAL_grib2.bin

※ **Z**と**C**の間はアンダースコアーは2個、その他のアンダースコアーは1個。

※ ***yyyyMMddhhmmss***はデータの解析時刻の年月日時分秒をUTCで設定。

4. ファイル形式

今回提供を開始するデータのファイル形式は、全て国際気象通報式FM92 GRIB 二進形式格子点資料気象通報式(第2版)(以下、「GRIB2」という)に則っています。GRIB2の詳細については国際気象通報式・別冊に詳しく記述されていますので、当該資料を参照願います。また、今回提供を開始するデータのGRIB2各節の詳細は、添付資料3を参照してください。

また、当該データの解読(デコード)処理については、添付資料4を参照してください。なお、サンプルデータは(財)気象業務支援センターに提供しておりますので、必要な場合は同センターへお問い合わせ下さい。

添付資料１

毎時大気解析GPVの概要

（１） データの概要

①解析時刻　：毎正時（1日24回）
②格子系　　：等緯度経度
③格子間隔　：地上は0.05度×0.0625度（格子数505×481）
④領域　　：（47.6N,120E)北西端、(22.4N,150E)を南東端とする領域
⑤データ量　：約18MB/回×24回＝432MB/日
⑥フォーマット：GRIB2

(2) データ内容（気圧面、要素）

気圧面 (hPa)	風	気温
地上	②	○
1000	②	○
975	②	○
950	②	○
925	②	○
900	②	○
850	②	○
800	②	○
700	②	○
600	②	○
500	②	○
400	②	○
300	②	○
250	②	○
200	②	○
150	②	○
100	②	○

＊②は2要素分のデータ（風の場合、東西方向と南北方向の2要素）
＊気温の配信は平成18年3月から行います。これ以降、気温に関する資料（GRIB2第4～7節）を追加します。

添付資料２

毎時大気解析の特徴（その1）風解析について

平成17年7月から配信を開始する毎時大気解析の風解析について、解析方法、解析例及び利用上の注意について説明します。なお、平成18年3月に開始する気温解析については、配信開始前に別途説明します。

毎時大気解析の風解析では、メソ予報モデル（MSM）の風予想値と各種観測データを利用して風の三次元分布の解析を行います。毎時の解析データを利用することで強風域や収束域の変化･移動を把握することができ、短時間の気象現象の予測に役立てることができます。

1. 解析の方法

解析には最適内挿法を使用し、MSMの風予想値を第一推定値とし、これをウィンドプロファイラ等の観測値で修整して作成します。なお、地上風は、高層風と独立して解析を行い、アメダス観測値で修整します。修整量は図1に示すように、

(1) 観測点での第一推定値と観測値との差をもとに、その点での修整量を決める。なお、観測の誤差が考慮されるため、修整後の値は観測値と同じ値にはならない。

(2) 各格子点における修整量を、観測点から離れるほど小さくなるように決める。

という方法で求めます。格子点の周囲に複数の観測値がある場合はそれぞれの修整量を加算し、観測値がない場合は修整しません。

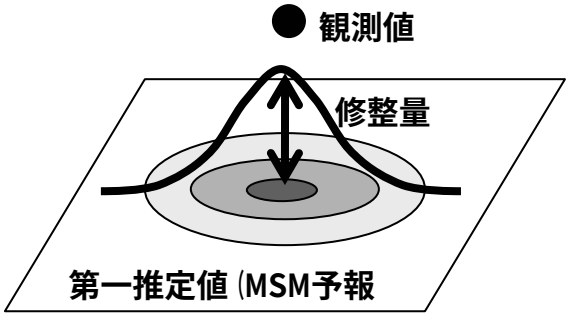


図1　修整量の決め方の模式図

2. 解析事例の紹介

平成16年6月30日、上空に寒気を伴ったトラフによる対流不安定な状態の中で、静岡県の駿河湾沿岸では内陸の冷気域と海上からの暖かい南西風により局地前線が形成され、その付近で強雨が継続しました。降水の最盛期にあたる9時の地上風を図2に示します。静岡から御前崎にかけての領域（図中の楕円）では、MSM予想値の風向が一様に南西となったのに対し、解析値では陸上の東北東風と太平洋からの南西風との収束が明瞭に表現できています。

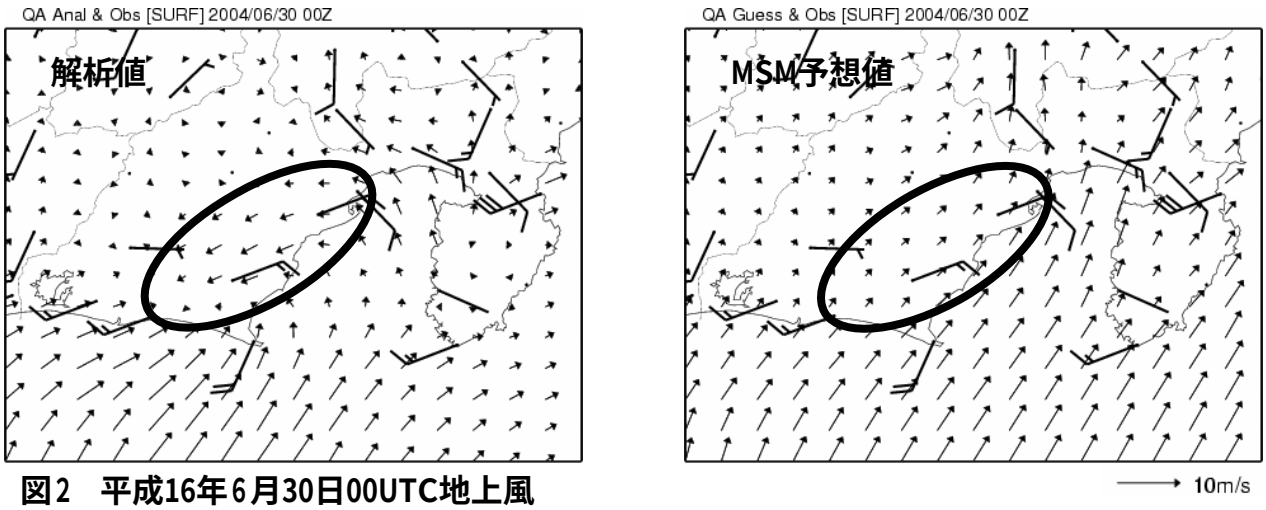


図2　平成16年6月30日00UTC地上風
左：解析値とアメダス観測値　右：MSM地上風予想値（注）とアメダス観測値
矢印：解析値またはMSM予想値、矢羽：アメダス観測値（長い棒が2m/s）
※当時の現業モデルは静力学MSMであるが、本図は非静力学MSMを用いて再計算した結果である。

3. 利用上の注意

(1) 風解析は全体的な風の空間分布や時間変化を把握するための資料です。雷雨等に伴う地上収束域や、メソスケール前線、上中層トラフの構造や動向などを理解する上で有用です。

(2)ただし、特に地上風では、地形の影響による局所的な風の強弱は平滑化され、解析値は数十kmスケールでの代表的な風の場となります。このため、特定の地点における風向･風速値としての利用には適しません。

(3) また、ダウンバーストや竜巻など、時間･空間スケールが小さい現象は表現できません。

(4) 観測点から離れた領域では修整量が小さく、解析値はMSM予想値とほぼ同じです。これは観測データがほとんどない海上や100hPa面で顕著です。

添付資料３

GRIB2通報式による
毎時大気解析格子点値
データフォーマット

平成17年４月

気象庁予報部

１．データについて

・データの範囲は、北緯22.4～47.6度、東経120～150度。
　経度方向の格子数は481、格子間隔は0.0625度、
　緯度方向の格子数は505、格子間隔は0.05度である。

・各フォーマット中のバイナリデータは、ビッグエンディアンである。

・負の値は最上位ビットを1にすることにより示す（2の補数表現ではない）

・単純圧縮において元のデータYは，次の式で復元できる。

$$Y = (R + X \times 2^{E}) \div 10^{D}$$

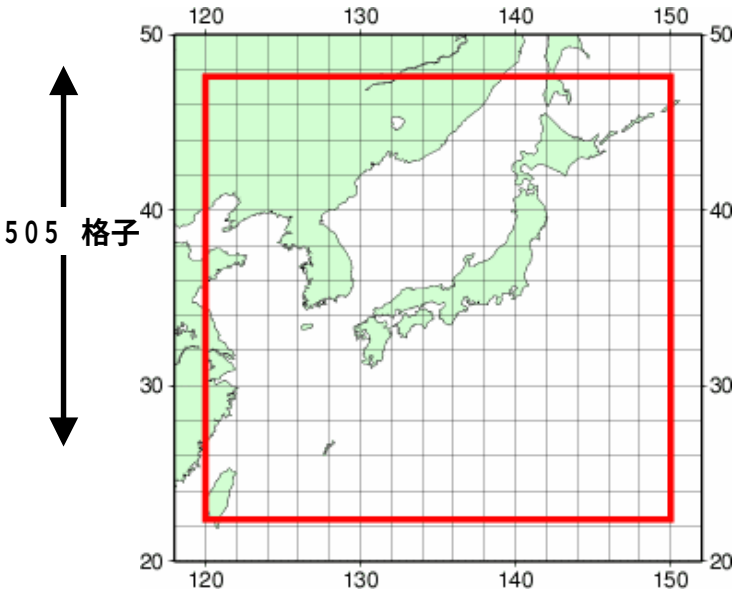
E＝二進尺度因子
D＝十進尺度因子
R＝参照値
X＝圧縮された値

注意事項

・要素、水平面の順序は不定である。

・平成18年3月から、気温データが追加される。

・GRIB2中の作成ステータスを利用して試験を行う場合があるので、
　必ず作成ステータス（第1節第20オクテット）を参照すること。



2．毎時大気解析に用いるＧＲＩＢ２のフォーマットおよびテンプレートの詳細

	節番号	節の名称・該当テンプレート	オクテット	内容	表	値	備考
	第０節	指示節	１～４	ＧＲＩＢ		"GRIB"	国際アルファベットNo.５(CCITT IA5)
			５～６	保留		missing	
			７	資料分野	符号表０．０	０	気象分野
			８	ＧＲＩＢ版番号		２	
			９～１６	ＧＲＩＢ報全体の長さ		******	12390529（気温なし），18585737（気温あり）
	第１節	識別節	１～４	節の長さ		２１	
			５	節番号		１	
			６～７	作成中枢の識別	共通符号表C-1	３４	東京
			８～９	作成副中枢		０	
			１０	ＧＲＩＢマスター表バージョン番号	符号表１．０	２	現行運用バージョン番号
			１１	ＧＲＩＢ地域表バージョン番号	符号表１．１	１	地域表バージョン１
			１２	参照時刻の意味	符号表１．２	０	解析
			１３～１４	資料の参照時刻（年）		******	
			１５	資料の参照時刻（月）		******	
			１６	資料の参照時刻（日）		******	
			１７	資料の参照時刻（時）		******	
			１８	資料の参照時刻（分）		******	
			１９	資料の参照時刻（秒）		******	
			２０	作成ステータス	符号表１．３	Ｔ	0=現業プロダクト、1=現業的試験プロダクト
			２１	資料の種類	符号表１．４	０	解析プロダクト
	第２節	地域使用節	不使用			省略	
	第３節	格子系定義節	１～４	節の長さ		７２	
			５	節番号		３	
			６	格子系定義の出典	符号表３．０	０	符号表３．１参照
			７～１０	資料点数		２４２９０５	５０５×４８１
			１１	格子点数を定義するリストのオクテット数		０	
			１２	格子点数を定義するリストの説明		０	
			１３～１４	格子系定義テンプレート番号	符号表３．１	０	緯度・経度格子
		ここからﾃﾝﾌﾟﾚｰﾄ3.0	１５	地球の形状	符号表３．２	６	半径6,371kmの球体と仮定した地球
		↓	１６	地球球体の半径の尺度因子		missing	
		↓	１７～２０	地球球体の尺度付き半径		missing	
		↓	２１	地球回転楕円体の長軸の尺度因子		missing	
		↓	２２～２５	地球回転楕円体の長軸の尺度付きの長さ		missing	
		↓	２６	地球回転楕円体の短軸の尺度因子		missing	
		↓	２７～３０	地球回転楕円体の短軸の尺度付きの長さ		missing	
		↓	３１～３４	緯線に沿った格子点数		４８１	
		↓	３５～３８	経線に沿った格子点数		５０５	
		↓	３９～４２	原作成領域の基本角		０	
		↓	４３～４６	端点の経度及び緯度並びに方向増分の定義に使われる基本角の細分		missing	
		↓	４７～５０	最初の格子点の緯度	10**-6度単位	４７６００００	北緯47.6度
		↓	５１～５４	最初の格子点の経度	10**-6度単位	１２００００００	東経120度
		↓	５５	分解能及び成分フラグ	フラグ表３．３	0x30	
		↓	５６～５９	最後の格子点の緯度	10**-6度単位	２２４００００	北緯22.4度
		↓	６０～６３	最後の格子点の経度	10**-6度単位	１５００００００	東経150度
		↓	６４～６７	i方向の増分	10**-6度単位	６２５００	0.0625度
		↓	６８～７１	j方向の増分	10**-6度単位	５００００	0.05度
		ここまでﾃﾝﾌﾟﾚｰﾄ3.0	７２	走査モード	フラグ表３．４	0x00	
要素および水平面毎に、第４節～第７節を繰り返す	第４節	プロダクト定義節	１～４	節の長さ		３４	
			５	節番号		４	
			６～７	テンプレート直後の座標値の数		０	
			８～９	プロダクト定義テンプレート番号	符号表４．０	０	ある時刻の，ある水平面における解析
		ここからﾃﾝﾌﾟﾚｰﾄ4.0	１０	パラメータカテゴリー	符号表４．１	※１	
		↓	１１	パラメータ番号	符号表４．２	※１	
		↓	１２	作成処理の種類	符号表４．３	０	解析
		↓	１３	背景作成処理識別符	ＪＭＡ定義	５１	毎時大気解析
		↓	１４	解析又は予報の作成処理識別符		missing	
		↓	１５～１６	観測資料の参照時刻からの締切時間（時）		０	
		↓	１７	観測資料の参照時刻からの締切時間（分）		２０	
		↓	１８	期間の単位の指示符	符号表４．４	１	時
		↓	１９～２２	予報時間		０	
		↓	２３	第一固定面の種類	符号表４．５	※２	
		↓	２４	第一固定面の尺度因子		※２	
		↓	２５～２８	第一固定面の尺度付きの値		※２	
		↓	２９	第二固定面の種類	符号表４．５	missing	
		↓	３０	第二固定面の尺度因子		missing	
		ここまでﾃﾝﾌﾟﾚｰﾄ4.0	３１～３４	第二固定面の尺度付きの値		missing	
	第５節	資料表現節	１～４	節の長さ		２１	
			５	節番号		５	
			６～９	全資料点の数		２４２９０５	５０５×４８１
			１０～１１	資料表現テンプレート番号	符号表５．０	０	格子点資料－単純圧縮
		ここからﾃﾝﾌﾟﾚｰﾄ5.0	１２～１５	参照値（R）（IEEE　32ビット浮動小数点）		R	Rは可変
		↓	１６～１７	二進尺度因子（E）		E	Eは可変
		↓	１８～１９	十進尺度因子（D）		D	Dは可変
		↓	２０	単純圧縮による各圧縮値のビット数		１２	
		ここまでﾃﾝﾌﾟﾚｰﾄ5.0	２１	原資料場の値の種類	符号表５．１	０	浮動小数点
	第６節	ビットマップ節	１～４	節の長さ		６	
			５	節番号		６	
			６	ビットマップ指示符		２５５	ビットマップを適用せず
	第７節	資料節	１～４	節の長さ		３６４３６３	
			５	節番号		７	
		ﾃﾝﾌﾟﾚｰﾄ7.0	６～nn	単純圧縮オクテット列		X～	単純圧縮された格子点値の列
	第８節	終端節	１～４	7777		"7777"	国際アルファベットNo.５(CCITT IA5)

（注）　値が「missing」の場合、そのデータは全ビット1の値、英数字の変数名や「******」は可変を示す。

※1 要素の表現 (第4節 10～11オクテットについて)

	10オクテット パラメータカテゴリ (符号表4．1)	11オクテット パラメータ番号 (符号表4．2)
風の東西成分	2 (運動量)	2 (風のu成分 m／s)
風の南北成分	2 (運動量)	3 (風のv成分 m／s)
気 温	0 (温度)	0 (温度 K)

※2 固定面の表現 (第4節 23～28オクテットについて)

	23オクテット 第一固定面の種類 (符号表4．5)	24オクテット 第一固定面の 尺度因子	25～28オクテット 第一固定面の 尺度付きの値
地上10m (風)	103 (地上からの特定高度面)	0	10
地上1.5m (気温)	103 (地上からの特定高度面)	1	15
1000 hPa	100 (等圧面 Pa)	-2	1000
975 hPa	〃	〃	975
950 hPa	〃	〃	950
925 hPa	〃	〃	925
900 hPa	〃	〃	900
850 hPa	〃	〃	850
800 hPa	〃	〃	800
700 hPa	〃	〃	700
600 hPa	〃	〃	600
500 hPa	〃	〃	500
400 hPa	〃	〃	400
300 hPa	〃	〃	300
250 hPa	〃	〃	250
200 hPa	〃	〃	200
150 hPa	〃	〃	150
100 hPa	〃	〃	100

添付資料４

毎時大気解析 GPV の解読処理サンプルプログラム

毎時大気解析GPVを解読処理するためのプログラムのサンプルを以下に示します。

本プログラムの全部又は一部を利用することは問題ありません。ただし、本プログラムの一部又は全部を利用したことにより、利用者が被った直接的または間接的ないかなる損害についても、気象庁は一切責任を負いません。本プログラムに関する個別の対応は行いかねますので、ご容赦願います。

sample_decoder.c

```
/*--------------------------------------------------------------------------
  毎時大気解析GPV GRIB2 ファイル サンプルデコード処理プログラム

    このプログラムの全部又は一部を利用してもかまいませんが、利用したこ
  とによって利用者が被った直接的又は間接的ないかなる損害についても、
  気象庁は一切責任を負いません。
    また、プログラムに関する個別の対応は行いかねますので、ご容赦願います。

  利用方法

  ANSI 準拠の C コンパイラでコンパイルして下さい。
  GRIB2 ファイルのファイル名を引数に与えて実行すると、
  ファイルの内容が表示されます。
--------------------------------------------------------------------------*/

# include <stdio.h>
# include <stdlib.h>
# include <math.h>


/* 1,2,4,8 : signed integer   (x byte)
          u : unsigned integer (1 byte)
          C : character        (4 byte)
          R : float            (4 byte) */

char *templateTable[9] = {
  "C2uu8",                          /* Section 0 */
  "4u22uuu2uuuuuuu",                /* Section 1 */
  "",                               /* Section 2 */
  "4uu4uu2uu4u4u4444444u4444u",     /* Section 3 */
  "4u22uuuuu2uu4u14uu4",            /* Section 4 */
  "4u42R22uu",                      /* Section 5 */
  "4uu",                            /* Section 6 */
  "4u",      /* + data */           /* Section 7 */
  "C"                               /* Section 8 */
};

```

```
38
39  static int isLittleEndian;
40
41  # define FIT_BYTE_ORDER(pointer,size) if(isLittleEndian)swabN(pointer,size,1)
42
43  /* reverse byte order */
44  static void
45  swabN( void *buf, int size, int nn )
46  {
47      char *ba, *bb, *buf2 = buf;
48      while( nn-- ) {
49          bb = ( ba = buf2 ) + size -1;
50          do {
51              char a;
52              a   = *ba;
53              *ba = *bb;
54              *bb =  a;
55          } while( ++ba < --bb );
56          buf2 += size;
57      }
58  }
59
60  int
61  read_section_0( FILE *fp, void **sec_buffer )
62  {
63      char *bufr;
64      *sec_buffer = realloc( *sec_buffer, 16 );
65      bufr = *sec_buffer;
66      if( fread( bufr, 1, 16, fp ) != 16 ||
67          strncmp( bufr, "GRIB", 4 ) != 0 ) {
68          fprintf( stderr, "Really GRIB file ?\n" );
69          exit(1);
70      }
71      return 0;
72  }
73
74  int
75  read_section_X( FILE *fp, void **sec_buffer )
76  {
77      int  length;
78      unsigned char *bufp;
79
80      fread( &length, 4, 1, fp );
81      if( strncmp( (char *)(&length), "7777", 4 ) == 0 ) {
82          *sec_buffer = realloc( *sec_buffer, 4 );
83          strncpy( *sec_buffer, "7777", 4 );
84          return 8;
85      }
86      FIT_BYTE_ORDER( &length, 4 );
87
88      *sec_buffer = realloc( *sec_buffer, length );
89      bufp = *sec_buffer;
90      if( fread( bufp + 4, 1, length - 4, fp ) != length - 4 ) {
91          fprintf( stderr, "Unexpected EOF\n" );
92          exit(1);
93      }
94      FIT_BYTE_ORDER( &length, 4 );
95      memcpy( bufp, &length, 4 );
96
97      return (int)bufp[4];   /* section No. */
98  }
99
```

```
100  void
101  decode_section( int secno, char *sec_buffer, double **double_values )
102  {
103      unsigned char buffer[8];
104              char *ttp;
105      int            size, ii, jj, missing;
106      double        *dvp, dd;
107
108      printf( "== SECTION %d == \n", secno );
109
110      ttp = templateTable[secno];
111
112      *double_values  =
113          realloc( *double_values, sizeof(double) * strlen(ttp) );
114
115      dvp = *double_values;
116
117      for( ii = 0 ; *ttp ; ttp++ ) {
118          switch( *ttp ) {
119              case '1' : case '2' : case '4' : case '8' :
120                  size = *ttp - '0';
121                  break;
122              case 'u' :
123                  size = 1;
124                  break;
125              case 'R' : case 'C' :
126                  size = 4;
127                  break;
128              default :
129                  fprintf( stderr, "Internal Error !\n" );
130                  exit(9);
131          }
132          memcpy( buffer, sec_buffer+ii, size );
133
134          missing = 1;
135          for( jj = 0 ; jj < size ; jj++ )
136              missing &= ( buffer[jj] == 0xFF );
137
138          switch( *ttp ) {
139              case 'u' : dd = *buffer;                        break;
140
141              case '1' :
142              case '2' :
143              case '4' :
144              case '8' : dd = buffer[0] & 0x7F;
145                         for( jj = 1 ; jj < size ; jj++ )
146                             dd = dd * 256.0 + buffer[jj];
147                         if( *buffer & 0x80 ) dd = -dd;       break;
148
149              case 'R' : FIT_BYTE_ORDER( buffer, size );
150                         dd = *(float *)buffer;               break;
151
152              case 'C' : memcpy( &dd, buffer, size );         break;
153          }
154          if( size == 1 )
155              printf( "   %4d     : ", ii+1 );
156          else
157              printf( "%4d .. %-4d: ", ii+1, ii+size );
158          if( missing )
159              printf( "missing \n" );
160          else if( *ttp == 'R' )
161              printf( "%f\n", dd );
```

```
162             else if( *ttp == 'C' )
163                 printf( "'%.4s'\n", &dd );
164             else
165                 printf( "%.0f\n", dd );
166 
167             *dvp++ = dd;
168             ii += size;
169         }
170     }
171 
172     void
173     unpack_data( const char *sec_7, const double *values_5, float **out )
174     {
175         double  rr, ee, dd, pow_2_e, pow_10_d;
176         float  *op;
177         int    num, nbit, ii;
178         unsigned int uw, mask;
179         const unsigned char *uc;
180 
181         uc = (const unsigned char *)sec_7 + 5;
182 
183         num   = values_5[2];
184         rr    = values_5[4];
185         ee    = values_5[5];
186         dd    = values_5[6];
187         nbit  = values_5[7];
188         pow_2_e  = pow( 2.0, ee);
189         pow_10_d = pow(10.0, dd);
190 
191         *out = realloc( *out, sizeof(**out) * num );
192         op = *out;
193 
194         mask = 1;
195         for( ii = 0 ; ii < nbit - 1 ; ii++ )
196             mask = mask << 1 | 1;
197 
198         for( ii = 0 ; ii < num ; ii++ ) {
199             memcpy( &uw, &uc[(nbit*ii)/8], 4 );
200             FIT_BYTE_ORDER( &uw, 4 );
201             uw = ( uw>>(32-nbit-(nbit*ii)%8)) & mask;
202             *op++ = ( rr + pow_2_e * uw ) / pow_10_d;
203         }
204     }
205 
206     # include <float.h>
207 
208     void
209     show_data_statistics( const double *values_4, const float *unpacked_data, int num )
210     {
211         int ii;
212         double sum = 0, min = DBL_MAX, max = -DBL_MAX;
213         char   level[256], *elem;
214 
215              if( values_4[4] == 0 && values_4[5] == 0 ) elem = "Temperature (K)";
216         else if( values_4[4] == 2 && values_4[5] == 2 ) elem = "U of wind (m/s)";
217         else if( values_4[4] == 2 && values_4[5] == 3 ) elem = "V of wind (m/s)";
218         else elem = "!!!!! Unknown Product Parameter !!!!!!";
219 
220              if( values_4[13] == 103 )
221                 sprintf( level, " %5.1f (m)  ",
222                                       values_4[15] / pow(10.0, values_4[14]) );
223         else if( values_4[13] == 100 )
```

```
224                 sprintf( level, " %5.0f (Pa) ",
225                                         values_4[15] / pow(10.0, values_4[14]) );
226         else    sprintf( level, "!!! Unknown Fixed surface type (%.0f) !!!",
227                                         values_4[13] );
228
229         printf( "  << %s %s >>\n", level, elem );
230
231         for( ii = 0 ; ii < num ; ii++ ) {
232             double ff = unpacked_data[ii];
233             sum  += ff;
234             if( min > ff ) min = ff;
235             if( max < ff ) max = ff;
236         }
237         printf( "  ( num = %d, min = %f, max = %f, average = %f )\n",
238                 num, min, max, sum / num );
239
240     }
241
242     void
243     save_float_file( const double *values_4, const float *unpacked_data, int num )
244     {
245         FILE *fpout;
246         char   fileName[256], *elem;
247
248              if( values_4[4] == 0 && values_4[5] == 0 ) elem = "T";
249         else if( values_4[4] == 2 && values_4[5] == 2 ) elem = "U";
250         else if( values_4[4] == 2 && values_4[5] == 3 ) elem = "V";
251         else elem = "Unknown_Product";
252
253              if( values_4[13] == 103 )
254                 sprintf( fileName, "%ssurf.dat", elem );
255         else if( values_4[13] == 100 )
256                 sprintf( fileName, "%s%.0f.dat",
257                     elem, values_4[15] / pow(10.0, values_4[14])/100.0 );
258         else    sprintf( fileName, "Unknown_Fixed_surface_type_(%.0f)",
259                     values_4[13] );
260
261         fprintf( stderr, "output '%s'\n", fileName );
262
263         if( ( fpout = fopen( fileName, "wb" ) ) == NULL ) {
264             fprintf( stderr, "file '%s' open error! \n", fileName );
265             exit(1);
266         }
267         fwrite( unpacked_data, sizeof(float), num, fpout );
268         fclose( fpout );
269     }
270
271     int main( int argc, char *argv[] )
272     {
273         FILE *fpin;
274         char *fileName;
275         int     secno, ii;
276         void   *sec_buffer = NULL;
277         float  *unpacked_data = NULL;
278         double *sec_double_value[9];
279
280         /* check system byte order */
281         isLittleEndian = 1;
282         isLittleEndian = *(char *)(&isLittleEndian);
283
284         for( ii = 0 ; ii < sizeof(sec_double_value)/sizeof(*sec_double_value) ;  )
285             sec_double_value[ii++] = NULL;
```

```
286         if( argc == 1 ) {
287              fprintf( stderr, "\n\n usage: %s 'grib2 file name'\n\n", argv[0] );
288              exit(1);
289         }
290         fileName = argv[1];
291         if( ( fpin = fopen( fileName, "rb" ) ) == NULL ) {
292             fprintf( stderr, "grib2 file '%s' open error! \n", fileName );
293             exit(1);
294         }
295         secno = read_section_0( fpin, &sec_buffer );
296         decode_section( secno, sec_buffer, &sec_double_value[secno] );
297
298         while( secno = read_section_X( fpin, &sec_buffer ) ) {
299             decode_section( secno, sec_buffer, &sec_double_value[secno] );
300             if( secno == 8 ) break;
301             if( secno == 7 ) {
302                 unpack_data( sec_buffer, sec_double_value[5], &unpacked_data );
303
304                 show_data_statistics( sec_double_value[4], unpacked_data,
305                                       (int)sec_double_value[5][2] );
306                 /*
307                 save_float_file ( sec_double_value[4], unpacked_data,
308                                       (int)sec_double_value[5][2] );
309                 */
310             }
311         }
312         fclose(fpin);
313         return 0;
314     }
```