（平成 21 年 8 月 20 日一部修正）
平 成 20 年 9 月 22 日
気象庁地球環境・海洋部

配信資料に関する技術情報（気象編） 第 286 号

～平成 21 年３月からの１か月予報・異常天候早期警戒情報関連の配信資料変更について～

（配信資料に関する技術情報（気象編） 第 205,246,274 号、
平成 20 年３月 10 日付お知らせ「１か月アンサンブル予報システムの変更」 関連）

毎週金曜日に、１か月予報資料（ファイル形式天気図画像）と１か月予報アンサンブル格子点値、１か月予報ガイダンスを配信しています。また、毎週火曜日に、異常天候早期警戒情報資料（ファイル形式天気図画像）を配信しています。

これら１か月予報・異常天候早期警戒情報関連の配信資料について、平成 21 年３月 13 日から変更を行います。変更の内容は、次の資料毎に【具体的な変更内容】１．～３．として説明します。

１．ファイル形式天気図画像
２．１か月予報アンサンブル格子点値
３．１か月予報ガイダンス

今回の変更により、「気候系監視年報」等の気象庁で用いる他の監視資料と同様の平年値（JRA-25 長期再解析に基づいた均質な平年値）やデータ作成方法を導入し、より充実した情報とします。また、予報システムの誤差特性を考慮する信頼度の高い確率ガイダンスの内容を拡充します。

【変更日時】

平成 21 年３月 13 日配信分（３月 12 日 12UTC 初期値の資料）から。

【具体的な変更内容】

１．ファイル形式天気図画像の変更点

(1) 大気循環場に関する平年値

大気循環場の偏差算出のため使用している、全球客観解析（GANAL）等に基づく平年値（1971 年～2000 年）について、JRA-25 長期再解析に基づいた長期間にわたり全球域で均質な平年値（1979 年～2004 年）を利用します。なお、JRA-25 長期再解析による解析値から求めた平年値の詳細については、配信資料に関する技術情報（気象編）第 246 号や気候系監視報告別冊第 13 号を参照してください。

(2) 平均循環場データの作成

これまで１日１回 12UTC の値から作成されていた平均循環場データを、１日４回（00，06，12，18UTC）の値に基づいて作成します。なお、データ作成方法の詳細については、配信資料に関する技術情報（気象編）第 246 号を参照してください。

(3) 200hPa 速度ポテンシャル時間経度断面図

各種時系列図を掲載する 1 か月予報資料（４）と異常天候早期警戒情報資料（３）には、5N-5S 平均 200hPa 速度ポテンシャル時間経度断面図を掲載しています。今後、これらの図の速度ポテンシャルの符号を、値の小さな領域から値の大きな領域に向かって発散風が吹くように定義します。この定義は「気候系監視年報」等の気象庁提供資料や国際的に用いられる多くの監視資料と同様です。

なお、発散が強い部分には引き続きハッチパターンとしますので、発散の強弱の予測の把握は同様に行えます。

(4) １か月予報資料の構成

現在、１か月予報資料（５）～（８）に全国の４週平均及び週別のガイダンスの値を掲載しています。１か月予報ガイダンスの変更（詳細は３．をご覧下さい）と合わせ、全国の４週平均の値を１か月予報資料（５）に、週別の値を１か月予報資料（６）に収めます。これに伴い、現在配信している 1 か月予報資料（７）及び（８）を廃止します。

詳細については、別紙「解説資料１」を参照してください。

２．１か月予報アンサンブル格子点値の変更点

(1) 大気循環場に関する平年値

１.（１）と同様

(2) 平均循環場データの作成

１.（２）と同様

(3) ファイルフォーマット

平年値や平均循環場データの作成方法の変更に伴い、フォーマットの一部仕様を変更するとともに、全てのファイルの形式を国際交換形式である GRIB2 に統一します。詳細については、別紙「解説資料２」を参照してください。

３．１か月予報ガイダンスおよびフォーマットの変更点

１か月予報ガイダンスの方式を１か月アンサンブル予報システムの誤差特性を考慮する信頼度の高い確率ガイダンスに一本化します。また、CSV 形式ファイルのフォーマットを詳細な確率分布が記述出来るものに変更します。これに伴い、これまで提供してきた日別予測式と期間平均予測式を廃止します。詳細については、別紙「解説資料３」を参照してください。

なお、確率ガイダンスの改良については、平成 20 年３月 10 日付のお知らせを参照して下さい。また、確率ガイダンスの詳細については、平成 20 年度季節予報研修テキスト（平成 20 年９月発行予定）に記述します。

この変更と同時期に、毎週火曜日に異常天候早期警戒情報ガイダンスの配信を新たに開始する予定です。詳細については後日お知らせにて連絡いたします。

解説資料１

平　成　２　０　年　９　月
気象庁地球環境・海洋部

１か月予報資料の構成変更について

１．変更概要

・　１か月予報資料（５）に全国の４週平均確率ガイダンス、１か月予報資料（６）に週別の確率ガイダンスを掲載します。
・　１か月予報資料（７）、（８）を廃止します。
・　総ページ数が８から６になります。

２．提供開始日

平成21年3月13日（金）から配信を予定しています。

３．１か月予報資料（５）～（８）の変更

１か月予報資料（５）～（８）を以下のように変更します。

掲載内容の新旧対照表

資料名	新 （3月１３日からの掲載内容）	現行 （3月６日までの掲載内容）	新しい構成例
１か月予報資料（５）	・全国の４週平均確率ガイダンス	・４週平均ガイダンス ・北日本、東日本の４週平均気温ヒストグラム	別添１－１
１か月予報資料（６）	・全国の１週目、２週目、３～４週目の確率ガイダンス	・４週平均ガイダンス ・西日本、沖縄・奄美の４週平均気温ヒストグラム	別添１－２
１か月予報資料（７）	廃止	・１週目、２週目、３～４週目のガイダンス ・全国の４週間降水量ヒストグラム	
１か月予報資料（８）	廃止	・１週目、２週目、３～４週目のガイダンス ・全国の４週間日照時間ヒストグラム	

新サンプル：１か月予報資料（５）

１か月予報資料（５）ガイダンス　　（４週間平均：１２月　８日～　１月　４日）　初期値：２００７年１２月　６日１２ＵＴＣ

期間平均予測式による確率ガイダンス

	気温　確率（%） 低　：　並　：　高	降水量　確率（%） 少　：　並　：　多	日照時間　確率（%） 少　：　並　：　多	降雪量　確率（%） 少　：　並　：　多	晴れ日数（日）　確率（%） 平年差（平年値）　少：多	降水日数（日）　確率（%） 平年差（平年値）　少：多	雨日数（日）　確率（%） 平年差（平年値）　少：多	
北日本	51: 22: 27	43: 37: 20	25: 30: 45	: :	-0.1(10.8) 45:55	-0.3(12.2) 60:40	0.1(2.4) 30:70	北日本
北日本日本海側	53: 20: 27	49: 38: 13	20: 41: 39	27: 43: 30	0.5(4.8) 45:55	-1.2(18.1) 60:40	0.2(3.7) 30:70	北日本日本海側
北日本太平洋側	49: 26: 25	37: 37: 26	36: 30: 34	: :	-0.7(16.0) 45:55	0.4(7.3) 60:40	0.4(1.0) 30:70	北日本太平洋側
東日本	34: 40: 26	39: 41: 20	30: 45: 25	: :	-0.1(17.1) 45:55	0.5(7.6) 60:40	0.4(2.8) 30:70	東日本
東日本日本海側	40: 39: 21	30: 47: 23	34: 38: 28	29: 21: 50	1.2(6.2) 45:55	-0.8(19.2) 60:40	0.1(8.6) 30:70	東日本日本海側
東日本太平洋側	35: 40: 25	42: 35: 23	29: 38: 33	: :	-0.7(20.0) 45:55	1.2(4.4) 60:40	0.8(1.4) 30:70	東日本太平洋側
北海道地方	45: 29: 26	46: 31: 23	19: 29: 52	: :	-0.2(10.3) 45:55	-0.8(13.7) 60:40	0.2(2.3) 30:70	北海道地方
北海道日本海側	49: 28: 23	49: 37: 14	18: 33: 49	36: 34: 30	0.8(4.8) 45:55	-1.7(18.7) 60:40	0.2(3.5) 30:70	北海道日本海側
北海道オホーツク海側	42: 30: 28	40: 38: 22	32: 37: 31	: :	-0.6(12.2) 45:55	-0.5(12.3) 60:40	0.0(1.5) 30:70	北海道オホーツク海側
北海道太平洋側	48: 29: 23	52: 26: 22	27: 32: 41	: :	-0.7(16.3) 45:55	0.0(8.3) 60:40	0.4(1.4) 30:70	北海道太平洋側
東北地方	38: 38: 24	29: 38: 33	29: 43: 28	: :	-0.3(11.8) 45:55	0.5(10.4) 60:40	0.2(2.4) 30:70	東北地方
東北日本海側	36: 41: 23	42: 34: 24	28: 40: 32	28: 36: 36	1.1(4.6) 45:55	-0.9(17.2) 60:40	0.2(4.3) 30:70	東北日本海側
東北太平洋側	43: 34: 23	30: 38: 32	32: 41: 27	: :	-0.6(16.9) 45:55	0.8(6.0) 60:40	0.4(1.0) 30:70	東北太平洋側
東北北部	43: 32: 25	41: 30: 29	31: 43: 26	: :	0.1(10.7) 45:55	0.3(11.8) 60:40	0.1(2.4) 30:70	東北北部
東北南部	37: 39: 24	30: 43: 27	34: 46: 20	: :	-0.3(12.8) 45:55	0.2(9.5) 60:40	0.3(2.5) 30:70	東北南部
関東甲信地方	41: 31: 28	46: 29: 25	29: 36: 35	: :	-0.9(20.7) 45:55	1.0(4.2) 60:40	0.7(1.2) 30:70	関東甲信地方
北陸地方	40: 39: 21	30: 47: 23	34: 38: 28	29: 21: 50	1.2(6.2) 45:55	-0.8(19.2) 60:40	0.1(8.6) 30:70	北陸地方
東海地方	32: 44: 24	32: 44: 24	29: 34: 37	: :	-0.2(19.1) 45:55	1.1(5.3) 60:40	0.5(1.6) 30:70	東海地方
西日本	28: 38: 34	28: 42: 30	40: 40: 20	: :	-0.1(14.5) 45:55	0.9(7.1) 60:40	0.5(2.1) 30:70	西日本
西日本日本海側	28: 39: 33	26: 49: 25	32: 43: 25	27: 28: 45	0.8(11.6) 45:55	0.6(9.2) 60:40	0.4(2.6) 30:70	西日本日本海側
西日本太平洋側	27: 34: 39	27: 41: 32	40: 43: 17	: :	-0.4(17.1) 45:55	0.9(5.4) 60:40	0.5(1.7) 30:70	西日本太平洋側
沖縄・奄美	14: 35: 51	18: 33: 49	43: 32: 25	: :	2.5(8.5) 45:55	0.0(10.6) 60:40	0.1(4.0) 30:70	沖縄・奄美
近畿地方	34: 40: 26	29: 41: 30	38: 44: 18	: :	-0.2(15.2) 45:55	0.5(7.3) 60:40	0.6(2.2) 30:70	近畿地方
近畿日本海側	34: 42: 24	29: 34: 37	42: 41: 17	35: 32: 33	0.1(9.6) 45:55	0.0(13.6) 60:40	0.7(4.2) 30:70	近畿日本海側
近畿太平洋側	34: 40: 26	31: 37: 32	39: 42: 19	: :	-0.4(17.3) 45:55	1.0(4.8) 60:40	0.4(1.5) 30:70	近畿太平洋側
中国地方	32: 37: 31	27: 43: 30	38: 39: 23	: :	0.5(12.0) 45:55	0.5(9.8) 60:40	0.8(3.0) 30:70	中国地方
山陰	32: 36: 32	21: 46: 33	41: 35: 24	33: 25: 42	1.0(8.0) 45:55	0.2(14.2) 60:40	0.8(4.7) 30:70	山陰
山陽	31: 34: 35	32: 40: 28	37: 31: 32	: :	0.0(16.9) 45:55	0.7(4.5) 60:40	0.5(1.1) 30:70	山陽
四国地方	26: 38: 36	29: 41: 30	37: 36: 27	: :	-0.2(17.8) 45:55	0.8(5.1) 60:40	0.5(1.6) 30:70	四国地方
九州北部地方	24: 37: 39	25: 46: 29	28: 51: 21	: :	0.2(13.2) 45:55	1.3(6.9) 60:40	0.7(1.6) 30:70	九州北部地方
九州南部・奄美地方	21: 34: 45	12: 41: 47	54: 31: 15	: :	0.2(14.9) 45:55	0.7(7.5) 60:40	0.5(2.5) 30:70	九州南部・奄美地方
九州南部	21: 37: 42	25: 42: 33	46: 37: 17	: :	-0.4(16.3) 45:55	1.1(6.8) 60:40	0.6(2.4) 30:70	九州南部
奄美地方	19: 41: 40	26: 36: 38	64: 28: 8	: :	0.5(8.5) 45:55	0.5(10.4) 60:40	0.1(3.6) 30:70	奄美地方
沖縄地方	7: 30: 63	21: 35: 44	42: 32: 26	: :	2.2(8.8) 45:55	-0.2(10.8) 60:40	0.2(4.0) 30:70	沖縄地方

新サンプル：１か月予報資料（６）

１か月予報資料（６）ガイダンス

初期値：2007 年12月 6日12ＵＴＣ

期間平均予測式による確率ガイダンス

（１週目気温：12 月 8 日～12 月14 日）

	気温 確率（%） 低 : 並 : 高		気温 確率（%） 低 : 並 : 高		気温 確率（%） 低 : 並 : 高		気温 確率（%） 低 : 並 : 高
北日本	26: 49: 25	北海道地方	29: 49: 22	西日本	2: 37: 61	近畿地方	6: 45: 49
北日本日本海側	27: 47: 26	東北地方	19: 51: 30	西日本日本海側	4: 39: 57	中国地方	7: 47: 46
北日本太平洋側	25: 47: 28	関東甲信地方	17: 44: 39	西日本太平洋側	2: 28: 70	四国地方	4: 41: 55
東日本	13: 46: 41	北陸地方	12: 51: 37	沖縄・奄美	0: 10: 90	九州北部地方	3: 35: 62
東日本日本海側	12: 51: 37	東海地方	9: 48: 43			九州南部・奄美地方	1: 27: 72
東日本太平洋側	15: 49: 36					沖縄地方	0: 11: 89

（２週目気温：12 月15 日～12 月21 日）

	気温 確率（%） 低 : 並 : 高	晴れ日数（日） 確率（%） 平年差（平年値） 少:多	降水日数（日） 確率（%） 平年差（平年値） 少:多
北日本	36: 32: 32	−0.5(2.8) 35:65	0.7(3.0) 75:25
北日本日本海側	43: 32: 25	−0.2(1.2) 35:65	0.4(4.6) 75:25
北日本太平洋側	33: 37: 30	−0.7(4.2) 35:65	0.7(1.7) 75:25
東日本	27: 42: 31	−0.1(4.4) 35:65	0.3(1.8) 75:25
東日本日本海側	35: 46: 19	0.1(1.4) 35:65	0.4(5.0) 75:25
東日本太平洋側	32: 37: 31	−0.1(5.2) 35:65	0.3(1.0) 75:25
北海道地方	33: 36: 31	−0.6(2.7) 35:65	0.7(3.3) 75:25
東北地方	34: 42: 24	−0.4(3.0) 35:65	0.7(2.6) 75:25
関東甲信地方	32: 39: 29	−0.1(5.4) 35:65	0.3(0.9) 75:25
北陸地方	35: 46: 19	0.1(1.4) 35:65	0.4(5.0) 75:25
東海地方	29: 39: 32	0.0(4.9) 35:65	0.3(1.2) 75:25
西日本	40: 36: 24	0.0(3.6) 35:65	0.4(1.6) 75:25
西日本日本海側	44: 36: 20	0.3(2.8) 35:65	0.5(2.1) 75:25
西日本太平洋側	39: 37: 24	0.0(4.3) 35:65	0.2(1.2) 75:25
沖縄・奄美	24: 40: 36	1.0(2.2) 35:65	−0.7(2.7) 75:25
近畿地方	31: 41: 28	−0.1(3.9) 35:65	0.1(1.8) 75:25
中国地方	46: 35: 19	0.0(3.0) 35:65	0.5(2.3) 75:25
四国地方	24: 38: 38	0.1(4.4) 35:65	0.1(1.1) 75:25
九州北部地方	35: 37: 28	0.0(3.2) 35:65	0.8(1.4) 75:25
九州南部・奄美地方	31: 37: 32	0.3(3.7) 35:65	0.1(1.7) 75:25
沖縄地方	19: 43: 38	0.8(2.3) 35:65	−0.8(2.7) 75:25

（３、４週目気温：12 月22 日～ 1 月 4 日）

	気温 確率（%） 低 : 並 : 高	晴れ日数（日） 確率（%） 平年差（平年値） 少:多	降水日数（日） 確率（%） 平年差（平年値） 少:多
北日本	44: 21: 35	0.2(5.4) 35:65	−0.4(6.0) 75:25
北日本日本海側	43: 22: 35	0.4(2.5) 35:65	−0.8(8.8) 75:25
北日本太平洋側	43: 25: 32	−0.1(8.0) 35:65	−0.2(3.7) 75:25
東日本	33: 35: 32	0.1(8.4) 35:65	0.0(3.9) 75:25
東日本日本海側	32: 35: 33	1.0(3.0) 35:65	−1.2(9.6) 75:25
東日本太平洋側	37: 31: 32	−0.4(9.9) 35:65	0.5(2.3) 75:25
北海道地方	41: 27: 32	0.1(5.2) 35:65	−0.6(6.8) 75:25
東北地方	40: 27: 33	0.0(6.0) 35:65	−0.2(5.1) 75:25
関東甲信地方	40: 30: 30	−0.6(10.3) 35:65	0.4(2.2) 75:25
北陸地方	32: 35: 33	1.0(3.0) 35:65	−1.2(9.6) 75:25
東海地方	37: 32: 31	−0.1(9.4) 35:65	0.3(2.8) 75:25
西日本	35: 32: 33	0.3(7.1) 35:65	−0.1(3.8) 75:25
西日本日本海側	41: 32: 27	0.7(5.6) 35:65	−0.4(4.9) 75:25
西日本太平洋側	35: 30: 35	0.0(8.4) 35:65	0.1(2.9) 75:25
沖縄・奄美	31: 35: 34	1.1(4.0) 35:65	−0.3(5.6) 75:25
近畿地方	39: 34: 27	0.2(7.3) 35:65	0.0(3.8) 75:25
中国地方	40: 32: 28	0.7(5.8) 35:65	−0.3(5.1) 75:25
四国地方	41: 29: 30	0.0(8.8) 35:65	0.2(2.8) 75:25
九州北部地方	34: 32: 34	0.6(6.5) 35:65	−0.3(3.8) 75:25
九州南部・奄美地方	31: 35: 34	0.2(7.2) 35:65	0.0(4.0) 75:25
沖縄地方	23: 39: 38	1.1(4.2) 35:65	−0.4(5.7) 75:25

解説資料２

平 成 ２ ０ 年 ９ 月
気象庁地球環境・海洋部

１か月予報アンサンブル格子点値について

１．変更概要

○１か月予報アンサンブル格子点値

標記データについて、その偏差算出のために使用する平年値を更新（全球客観解析（GANAL）等に基づく平年値→JRA-25 長期再解析に基づく平年値）するとともに、平均循環場データの作成方法を変更（12UTC の値→１日４回（00，06，12,18UTC）の値に基づいて作成）します。また、これらの変更に伴い、フォーマットの一部仕様を変更するとともに、ファイルの形式を GRIB2 に統一します。変更点は以下のとおりです。配信ファイル形式の詳細については、別添２－１を参照してください。

①１か月予報メンバー別全球格子点値

・平均循環場データの作成方法の変更
（12UTC の値→１日４回（00，06，12,18UTC）の値を使用）
・GRIB2 ビット数の変更（12 ビット→16 ビット）
・GRIB2 第４節定義節の変更（テンプレート 4.1→テンプレート 4.11）
・降水量の単位の変更（初期値からの積算降水量(㎜)→予報対象日の日降水量（㎜/day））
・予報時間の変更（34 日→33 日）※
・データ量（1GB→1.3GB）

※ 18UTC の値がない予報最終日（34 日目）は、１日４回（00，06，12,18UTC）に基づく値を作成できないため。

②１か月予報アンサンブル統計全球格子点値※（現：１か月予報アンサンブル統計格子点値）

・平均循環場データの作成方法の変更
（12UTC の値→１日４回（00，06，12,18UTC）の値を使用）
・偏差算出のために使用する平年値の変更（GANAL 等→JRA-25 に基づく平年値）
・ファイル形式の変更（GRIB1→GRIB2（16 ビット））
・全要素について全球データを配信
・降水量の単位の変更（期間の総降水量(㎜)→ 期間の日平均降水量（㎜/day））
・ファイル名の変更（気象庁ファイル命名規則に従ったファイル名に変更）

変更前：Z__C_RJTD_yyyyMMddhhmmss_EPS1_GPV_Rgl_Eem_grib.bin

変更後：Z__C_RJTD_yyyyMMddhhmmss_EPS1_GPV_Rgl_Eem_grib2.bin

・データ量（1MB→1.8MB）

※②の名称は、全要素について全球データを配信することから、現在の（）内の名称から変更します。

２．提供開始日

平成 21 年３月 13 日配信分（３月 12 日 12UTC 初期値の予報）から。

３． サンプルデータ

サンプルデータを媒体（DVD-RW　１枚）に収録して気象業務支援センターから提供しますので、必要な場合はご利用下さい。

１）サンプルデータを収録した１枚の DVD－RW の構成について

フォルダ名	格納ファイル
GDC	１か月予報ガイダンス（35 ファイル）及び同 tar 結合ファイル
ＧＰＶ	１か月予報アンサンブル統計全球格子点値（１ファイル）
MGPV	１か月予報メンバー別全球格子点値（要素別 36 ファイル）
program	１か月予報アンサンブル統計全球格子点値および１か月予報メンバー別全球格子点値解読処理用サンプルプログラム（Windows 実行形式ファイル）

２）解読（デコード）処理について

１か月予報アンサンブル格子点値は解読（デコード）処理が必要です。データの解読処理については、別添２－２および２－３をそれぞれ参照してください。また、参考までに解読処理用サンプルプログラムの Windows 実行形式ファイルを上記のサンプルデータとともに DVD-RW に収録しましたので、ご利用ください。

解説資料３

平　成　２　０　年　９　月
気象庁地球環境・海洋部

１か月予報ガイダンスの変更について

１．変更概要

・ 予測式を確率ガイダンスに一本化（日別予測式、期間平均予測式を廃止）し、CSV 形式ファイルのフォーマットを確率ガイダンスに相応しいものへ変更する。
・ 地点のガイダンス値を新たに配信する。
・ 総データ容量が約 20MB から約 60MB に増加する。

２．提供開始日

平成 21 年３月 13 日（金）からの配信を予定。

３．CSV 形式ファイルの命名規則

地点のファイル名は次のとおりである。
Z__C_RJTD_*yyyyMMddhhmmss*_EPS1_GUID_RS-all_JRlong_P-all_tablr.csv
地域平均のファイル名は、従来から変更がなく、下表にある地域略称を用いて、次のとおりである。
Z__C_RJTD_*yyyyMMddhhmmss*_EPS1_GUID_RJ（地域略称）_JRlong_P-all_tablr.csv
ここで、*yyyyMMddhhmmss* は初期時刻の年月日時分秒を UTC（世界協定時）で示す。

地域略称と地域の対応表

略称	地域名	略称	地域名	略称	地域名
knh00	北日本	hkd00	北海道地方	knk00	近畿地方
knh01	北日本日本海側	hkd01	北海道日本海側	knk01	近畿日本海側
knh02	北日本太平洋側	hkd03	北海道オホーツク海側	knk02	近畿太平洋側
hnh00	東日本	hkd02	北海道太平洋側	cgk00	中国地方
hnh01	東日本日本海側	thk00	東北地方	cgk06	山陰
hnh02	東日本太平洋側	thk01	東北日本海側	cgk07	山陽
nnh00	西日本	thk02	東北太平洋側	skk00	四国地方
nnh01	西日本日本海側	thk05	東北北部	kyh00	九州北部地方
nnh02	西日本太平洋側	thk04	東北南部	kyn00	九州南部・奄美地方
nss00	沖縄・奄美	ktk00	関東甲信地方	kyn08	九州南部
		hkr00	北陸地方	kyn09	奄美地方
		tki00	東海地方	okn00	沖縄地方

４．サンプルデータ

サンプルデータを用意する。利用方法については、解説資料２の３．サンプルデータを参照下さい。

５．CSV 形式ファイルのフォーマットの変更

CSV 形式（カンマで区切られたテキストデータ）ファイルのフォーマット（１行の構成と行の並び方）を以下のように変更する。変更点はメンバー毎の値を廃止して、累積確率値を格納する。また、地域平均値については５階級区分別の確率と区分値も格納する。

①　行の構成

各ファイルは「初期時刻行」、「予測資料行」の２種の行により構成されている。各行の説明を下記に示

す。表の１段目はカラムの説明、２段目は文字数、３段目は数値予報ガイダンスのデータの例を示す。各カラムにデータを右詰で収録し、余りはスペースで埋める。各表の下にカラムの詳細を示した。各表中の「C」はカンマ（,）のカラムを表す。

初期時刻行　（変更後）

初期値年	C	初期値月	C	初期値日	C	バージョン情報	C
5	1	5	1	5	1	5	1
2005	,	9	,	1	,	1	,

初期値年・月・日・時・分は、ガイダンスの基である数値予報モデルの初期値時刻を世界標準時を用いて表す。
バージョン情報は、今回の変更から付加する。フォーマットの変更等に応じた通し番号である。

地域平均の予測資料行　（変更後）

予測対象期間開始年	C	予測対象期間開始月	C	予測対象期間開始日	C	予測対象期間終了年	C	予測対象期間終了月	C	予測対象期間終了日	C	予測対象期間長	C	
5	1	5	1	5	1	5	1	5	1	5	1	5	1	続く
2005	,	8	,	26	,	2005	,	9	,	1	,	7	,	

地域番号	C	要素番号	C	予測式の種別	C	予測値*（アンサンブル平均値)	C	累積確率(閾値１）*	C	…	C	累積確率(閾値101）*	C	かなり低(少な）い確率*	C	
5	1	5	1	5	1	5	1	5	1	…	1	5	1	5	1	続く
1	,	1	,	3	,	10	,	0	,	…	,	100	,	5	,	

低（少な）い確率*	C	平年並の確率*	C	高（多）い確率*	C	かなり高(多)い確率*	C	かなり低（少な）い区分値*	C	低（少な）い区分値*	C	高（多）い区分値*	C	かなり高（多）い区分値*	C
5	1	5	1	5	1	5	1	5	1	5	1	5	1	5	1
15	,	25	,	35	,	20	,	-20	,	-8	,	8	,	15	,

地域平均の予測資料行　（変更前）

予測対象期間開始年	C	予測対象期間開始月	C	予測対象期間開始日	C	予測対象期間終了年	C	予測対象期間終了月	C	予測対象期間終了日	C	予測対象期間長	C	
5	1	5	1	5	1	5	1	5	1	5	1	5	1	続く
2005	,	8	,	26	,	2005	,	9	,	1	,	7	,	

地域番号	C	要素番号*	C	予測式の種別*	C	予測値*(アンサンブル平均)	C	予測値*(メンバー1)	C	予測値*(メンバー2)	C	…	C	予測値*(メンバー50)	C
5	1	5	1	5	1	5	1	5	1	5	1	…	1	5	1
1	,	1	,	1	,	-7	,	-4	,	-10	,	…	,	-6	,

地点の予測資料行　（新規）

予測対象期間開始年	C	予測対象期間開始月	C	予測対象期間開始日	C	予測対象期間終了年	C	予測対象期間終了月	C	予測対象期間終了日	C	予測対象期間長	C	
5	1	5	1	5	1	5	1	5	1	5	1	5	1	続く
2005	,	8	,	26	,	2005	,	9	,	1	,	7	,	

地点番号	C	要素番号	C	予測式の種別	C	予測値*（アンサンブル平均値)	C	累積確率(閾値１）*	C	…	C	累積確率(閾値141）*	C
5	1	5	1	5	1	5	1	5	1	…	1	5	1
1	,	1	,	3	,	10	,	0	,	…	,	100	,

予測対象期間長　：本行が収録している予測結果の対象期間の長さを日を単位として表す。

地域番号　　：表１にまとめた。
要素番号、予測値：予測要素に割り振った番号と予測値の単位について表２にまとめた。
予測式の種別　：表３にまとめた。今回の変更により、予測式の種別は「３」のみとなる。
確率値　　：天気日数については、「平年値より少ない確率」と「平年値以上の確率」となる。
区分値　　：地域平均階級区分値として表４にまとめた。
累積確率　　：閾値の範囲を表５にまとめた。
地点番号　　：表６にまとめた。
地点の累積確率　：地点の閾値の範囲を表７にまとめた。

*がついているカラムは値が無い場合"-9999"とする。

② 行の並び

各ファイル中のデータ行の並びは次のとおり。ただし、これまで格納していた初期値日の６日前からの実況値分は省く。

１か月予報ガイダンス

第１行：「初期時刻行」
第２行以降は、「予測資料行」を次のとおり並べる。
確率ガイダンスについて、要素番号1～7 ごとに予測対象期間長７日、14 日、28 日の順に並べる。

表１　地域番号と地域名　（変更なし）

番号	地域名	番号	地域名	番号	地域名
1	北日本	11	北海道地方	23	近畿地方
2	北日本日本海側	12	北海道日本海側	24	近畿日本海側
3	北日本太平洋側	13	北海道オホーツク海側	25	近畿太平洋側
4	東日本	14	北海道太平洋側	26	中国地方
5	東日本日本海側	15	東北地方	27	山陰
6	東日本太平洋側	16	東北日本海側	28	山陽
7	西日本	17	東北太平洋側	29	四国地方
8	西日本日本海側	18	東北北部	30	九州北部地方
9	西日本太平洋側	19	東北南部	31	九州南部・奄美地方
10	沖縄・奄美	20	関東甲信地方	32	九州南部
		21	北陸地方	33	奄美地方
		22	東海地方	34	沖縄地方

表２　予測要素とアンサンブル平均値の単位　（変更なし）

要素番号	要素	単位
1	気温平年差	0.1℃
2	降水量平年比	%
3	日照時間平年比	%
4	晴れ日数*1	0.1 日
5	降水日数*2	0.1 日
6	雨日数*3	0.1 日
7	降雪量平年比	%

*1　日照率40%以上の日数。日照率は、１日の日照時間を
　　可照時間（日の出から日の入りまでの時間）で割った値。
*2　日降水量１㎜以上の日数。
*3　日降水量10㎜以上の日数。

表３　予測式の種別　（変更なし）

番号	意味
1	日別予測式
2	期間平均予測式
3	確率ガイダンス

表４　階級区分値　（新規）

項目	意味
「かなり低（少な）い」の区分値	この値以下でかなり低（少な）い階級となる
「低（少な）い」の区分値	この値以下で低（少な）い階級となる
「高（多）い」の区分値	この値より大きいと高（多）い階級となる
「かなり高（多）い」の区分値	この値より大きいとかなり高（多）い階級となる

表５　閾値の範囲　（新規）

要素	閾値１	閾値２	…	閾値 51	…	閾値 100	閾値 101	増分
気温平年差（0.1℃）	-50	-49		0		49	50	1
降水量平年比（%）	0	2		100		198	200	2
日照時間平年比（%）	0	2		100		198	200	2
晴れ日数平年差（0.1 日）	-50	-49		0		49	50	1
降水日数平年差（0.1 日）	-50	-49		0		49	50	1
雨日数平年差（0.1 日）	-50	-49		0		49	50	1
降雪量平年比（%）	0	2		100		198	200	2

表6　国際地点番号と地点名　（新規）

47401	稚　内	47602	相　川	47677	三宅島	47821	阿蘇山
47402	北見枝幸	47604	新　潟	47678	八丈島	47822	延　岡
47404	羽　幌	47605	金　沢	47682	千　葉	47823	阿久根
47405	雄　武	47606	伏　木	47684	四日市	47824	人　吉
47406	留　萌	47607	富　山	47690	日　光	47827	鹿児島
47407	旭　川	47610	長　野	47740	西　郷	47829	都　城
47409	網　走	47612	高　田	47741	松　江	47830	宮　崎
47411	小　樽	47615	宇都宮	47742	境	47831	枕　崎
47412	札　幌	47616	福　井	47744	米　子	47835	油　津
47413	岩見沢	47617	高　山	47746	鳥　取	47836	屋久島
47417	帯　広	47618	松　本	47747	豊　岡	47837	種子島
47418	釧　路	47620	諏　訪	47750	舞　鶴	47838	牛　深
47420	根　室	47622	軽井沢	47754	萩	47843	福　江
47421	寿　都	47624	前　橋	47755	浜　田	47887	松　山
47423	室　蘭	47626	熊　谷	47756	津　山	47890	多度津
47424	苫小牧	47629	水　戸	47759	京　都	47891	高　松
47426	浦　河	47631	敦　賀	47761	彦　根	47892	宇和島
47428	江　差	47632	岐　阜	47762	下　関	47893	高　知
47430	函　館	47636	名古屋	47765	広　島	47895	徳　島
47433	倶知安	47637	飯　田	47766	呉	47897	宿　毛
47435	紋　別	47638	甲　府	47767	福　山	47898	清　水
47440	広　尾	47640	河口湖	47768	岡　山	47899	室戸岬
47512	大船渡	47641	秩　父	47769	姫　路	47909	名　瀬
47520	新　庄	47646	館　野	47770	神　戸	47912	与那国島
47570	若　松	47648	銚　子	47772	大　阪	47918	石垣島
47574	深　浦	47649	上　野	47776	洲　本	47927	宮古島
47575	青　森	47651	津	47777	和歌山	47929	久米島
47576	む　つ	47653	伊良湖	47778	潮　岬	47936	那　覇
47581	八　戸	47654	浜　松	47780	奈　良	47940	名　護
47582	秋　田	47655	御前崎	47784	山　口	47942	沖永良部
47584	盛　岡	47656	静　岡	47800	厳　原	47945	南大東島
47585	宮　古	47657	三　島	47805	平　戸	47971	父　島
47587	酒　田	47662	東　京	47807	福　岡		
47588	山　形	47663	尾　鷲	47809	飯　塚		
47590	仙　台	47666	石廊崎	47812	佐世保		
47592	石　巻	47668	網　代	47813	佐　賀		
47595	福　島	47670	横　浜	47814	日　田		
47597	白　河	47672	館　山	47815	大　分		
47598	小名浜	47674	勝　浦	47817	長　崎		
47600	輪　島	47675	大　島	47819	熊　本		

表７　地点の閾値の範囲　（新規）

要素	閾値１	閾値２	…	閾値 71	…	閾値 140	閾値 141	増分
気温平年差（0.1℃）	-70	-69		0		69	70	1
降水量平年比（%）	0	2		140		278	280	2
日照時間平年比（%）	0	2		140		278	280	2
晴れ日数平年差（0.1 日）	-70	-69		0		69	70	1
降水日数平年差（0.1 日）	-70	-69		0		69	70	1
雨日数平年差（0.1 日）	-70	-69		0		69	70	1
降雪量平年比（%）	0	2		140		278	280	2

１か月予報アンサンブル格子点値の解説

１．概要

１か月予報アンサンブル格子点値には、１か月予報メンバー別全球格子点値と１か月予報アンサンブル統計全球格子点値がある。

（１）１か月予報メンバー別全球格子点値（日別）

作成回数　　：　週１回
予報時間　　：　33 日間（１日間間隔）
ｱﾝｻﾝﾌﾞﾙﾒﾝﾊﾞｰ数：　50 ﾒﾝﾊﾞｰ（水曜日 25 ﾒﾝﾊﾞｰ、木曜日 25 ﾒﾝﾊﾞｰ合計 50 ﾒﾝﾊﾞｰ）
格子系　　：　等緯度経度（2.5 度格子）
領域　　：　全球
データ内容　：

地上要素

	海面更正気圧*	積算降水量	２ｍ気温*
地上	○	○	○

気圧面要素

気圧面 (hPa)	高度*	風	気温*	相対湿度
1000	○	◎	○	○
850	○	◎	○	○
700	○	◎	○	○
500	○	◎	○	○
300	○	◎	○	○
200	○	◎	○	
100	○	◎	○	

◎東西方向と南北方向の２要素
*海面更正気圧、高度、気温は系統誤差補正済み。

（２）１か月予報アンサンブル統計全球格子点値

作成回数　：　週１回
予報期間　：　４週間（１週間間隔または２週間間隔）
統計処理　：
　　（ﾒﾝﾊﾞｰ）アンサンブル平均
　　（期間）１週間平均、２週間平均または４週間平均
格子系　　：　等緯度経度（2.5 度格子）
領域　　：　全球
データ内容：

地上要素

	海面更正気圧	海面更正気圧平年偏差	積算降水量
地上	○	○	○

気圧面要素

気圧面 (hPa)	高度	高度平年偏差	風	気温	気温平年偏差	相対湿度	その他
850			◎	○	○	○	気温スプレッド*1
500	○	○					高度スプレッド 高度高偏差確率*2
200			◎				
100	○	○					

◎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）

統計処理、予報の時間間隔は、要素により異なっている、詳細な内容は以下のとおり。

要素		レベル (hPa)	領域	予報対象期間
ｱﾝｻﾝﾌﾞﾙ平均値（7 日平均値場）	海面更正気圧、積算降水量	-	全球 2.5x2.5 度	予報初日から 2-8, 9-15, 16-22, 23-29 日
	気温、相対湿度、風(東西成分、南北成分)	850		
	ｼﾞｵﾎﾟﾃﾝｼｬﾙ高度	500,100		
	風(東西成分、南北成分)	200		
	海面更正気圧の平年偏差	-		
	気温の平年偏差	850		
	ｼﾞｵﾎﾟﾃﾝｼｬﾙ高度の平年偏差	500,100		
ｱﾝｻﾝﾌﾞﾙﾒﾝﾊﾞｰ間のｽﾌﾟﾚｯﾄﾞ	海面更正気圧	-		予報初日から 2-8, 9-15, 16-22, 23-29,2-15, 16-29, 2-29 日
	気温	850		
	ｼﾞｵﾎﾟﾃﾝｼｬﾙ高度	500		
高偏差確率.		500		

＊１ スプレッド　：　予報メンバの標準偏差を自然変動の標準偏差で規格化した値。

＊２ 高偏差確率　：　予想される偏差の絶対値が自然変動の標準偏差の 0.5 倍を上回る確率。

２．ファイルフォーマット等の詳細

（１）１か月予報アンサンブル統計全球格子点値
ファイル名　　：「１か月予報アンサンブル格子点値ファイル名」参照
レコード形式　：「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）（GRIB２）」
「１か月予報アンサンブル統計全球格子点値ファイルにおける GRIB2 第４節の補足説明（別紙１）」参照
ファイルサイズ：1.8MB（予報日時（期間）別，領域別，層別，物理量別に格納）

（２）１か月予報メンバー別全球格子点値
ファイル名　　：「１か月予報アンサンブル格子点値ファイル名」参照
レコード形式　：「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）（GRIB２）」
「１か月予報メンバー別全球格子点値ファイルにおける GRIB2 第４節の補足説明（別紙２）」参照
ファイルサイズ：１ファイルあたり 36 MB、36 ファイルの合計約 1.3GB

１か月予報アンサンブル格子点値ファイル名

	ファイル名称	サイズ (MB)	データ内容
１か月予報メンバー別全球格子点値	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lsurf_Ppp_Emb_grib2.bin	36	海面更正気圧
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lsurf_Prr_Emb_grib2.bin	36	日降水量
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lh2_Ptt_Emb_grib2.bin	36	2m 気温
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp1000_Phh_Emb_grib2.bin	36	1000hPa 高度
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp1000_Pwu_Emb_grib2.bin	36	1000hPa 風（東向き成分）
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp1000_Pwv_Emb_grib2.bin	36	1000hPa 風（北向き成分）
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp1000_Ptt_Emb_grib2.bin	36	1000hPa 気温
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp1000_Prh_Emb_grib2.bin	36	1000hPa 相対湿度
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp850_Phh_Emb_grib2.bin	36	850hPa 高度
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp850_Pwu_Emb_grib2.bin	36	850hPa 風（東向き成分）
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp850_Pwv_Emb_grib2.bin	36	850hPa 風（北向き成分）
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp850_Ptt_Emb_grib2.bin	36	850hPa 気温
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp850_Prh_Emb_grib2.bin	36	850hPa 相対湿度
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp700_Phh_Emb_grib2.bin	36	700hPa 高度
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp700_Pwu_Emb_grib2.bin	36	700hPa 風（東向き成分）

	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp700_Pwv_Emb_grib2.bin	36	700hPa 風（北向き成分）
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp700_Ptt_Emb_grib2.bin	36	700hPa 気温
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp700_Prh_Emb_grib2.bin	36	700hPa 相対湿度
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp500_Phh_Emb_grib2.bin	36	500hPa 高度
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp500_Pwu_Emb_grib2.bin	36	500hPa 風（東向き成分）
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp500_Pwv_Emb_grib2.bin	36	500hPa 風（北向き成分）
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp500_Ptt_Emb_grib2.bin	36	500hPa 気温
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp500_Prh_Emb_grib2.bin	36	500hPa 相対湿度
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp300_Phh_Emb_grib2.bin	36	300hPa 高度
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp300_Pwu_Emb_grib2.bin	36	300hPa 風（東向き成分）
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp300_Pwv_Emb_grib2.bin	36	300hPa 風（北向き成分）
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp300_Ptt_Emb_grib2.bin	36	300hPa 気温
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp300_Prh_Emb_grib2.bin	36	300hPa 相対湿度
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp200_Phh_Emb_grib2.bin	36	200hPa 高度
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp200_Pwu_Emb_grib2.bin	36	200hPa 風（東向き成分）
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp200_Pwv_ Emb_grib2.bin	36	200hPa 風（北向き成分）
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp200_Ptt_Emb_grib2.bin	36	200hPa 気温
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp100_Phh_Emb_grib2.bin	36	100hPa 高度
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp100_Pwu_Emb_grib2.bin	36	100hPa 風（東向き成分）
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp100_Pwv_ Emb_grib2.bin	36	100hPa 風（北向き成分）
	Z__C_RJTD_yyyyMMddhhmmss_EPS1_MGPV_Rgl_Lp100_Ptt_Emb_grib2.bin	36	100hPa 気温
１か月予報アンサンブル統計全球格子点値	Z__C_RJTD_yyyyMMddhhmmss_EPS1_GPV_Rgl_Eem_grib2.bin	1.8	アンサンブル統計値

全ﾌｧｲﾙ数 ＝ 36+1 ファイル

（１か月予報メンバー別全球、36 要素（36 ﾌｧｲﾙ）＋ １か月予報アンサンブル統計全球（１ﾌｧｲﾙ））

このファイル名は、国際的な資料交換に用いるため、世界気象機関（WMO）により採用されたファイル命名規則に準拠し、任意部分を当庁において定義したものである（技術情報第 130 号）。

Z__C　　：　Z と C の間には、アンダースコア“_”が２つ続く“__”

yyyyMMddhhmmss　　：　数値予報の初期値年月日時を表す。mmss は 0000 とする。

用語説明

・アンサンブル予報：観測（解析）誤差程度のわずかな違いのある複数の初期値をもとに数値予報を行ない、それぞれの結果を統計的に処理する予測手法。

・メンバー　　：アンサンブル予報を構成している個々の予報。

・アンサンブル平均：各メンバーを平均して求めた予測結果。

・スプレッド　　：予報メンバの標準偏差を自然変動の標準偏差で規格化した値。アンサンブル予報を構成しているメンバー間のばらつきの大きさを示す指標。

・高偏差確率　　：予想される偏差の絶対値が自然変動の標準偏差の 0.5 倍を上回る確率。

1か月予報メンバー別全球格子点値ファイルの解読（デコード）処理について

１．解読サンプルプログラムのソースコード
別紙参照。

２．利用方法
以下、解読サンプルプログラムの
ソースコードのファイル名：dcd_1megrib2m_sample.c
実行ファイル名　　：dcd_1merib2m
です。

（１）ccコマンドによりコンパイルしてください。その際標準算術関数を利用可能なようにライブラリをリンクしてください。
実行例）　$ cc dcd_1megrib2m_sample.c – lm – o dcd_1merib2m
(dcd_1megrib2m と言う実行ファイルが生成される)

- ANSI 準拠の c コンパイラでコンパイルできます。UNIX（HP-UX）及び Linux（RedHat）での動作を確認しています。
- リトルエンディアンマシンにも対応しています。

（２）次のコマンドを入力することにより、１か月予報メンバー別全球格子点値,各節の内容が端末に表示されると共に、４バイト実数形式でデコードされたデータがファイルに書き出されます。
実行例）$ dcd_1megrib2m　{１か月予報メンバー別全球格子点値ファイル名}

- デコードされたデータは、予報時間ごと、メンバーごとに、サンプルプログラムで割り付けた複数のファイルに出力されます。ファイル名は159行で使用されている関数identifydata 1269～1282行で割り付けています。
- ファイルの出力を止めたい場合は、サンプルプログラム14行目の
  #define IFOUT　1
  を
  #define IFOUT　0
  など、1以外の数字に変更してください。

別紙

解読サンプルプログラムのソースコード
（１か月予報メンバー別全球格子点値ファイル用）

```
/*****************************************************************************
**   Sample Program to convert   1 month   ensemble forecast GPV(GRIB2) of JMA
**                                          2008.07     JMA/CPD
**
**              Tested on Windows (gcc ,borlandc), Linux (gcc)
**                        and HP-UX (native c compiler)
*****************************************************************************/
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <ctype.h>

#define IFOUT           1           /*   1 :Output decoded data                */
#define MASK7           0x7F
#define MASK1           0x80
#define UNDEF           -19999.
#define SIZEOFBUF       10000
#define SIZEOFBITBUF 50000
#define MAX_BUFF        50000       /* buffer size for gpv MAX_BUFF < 4294967295   2^32 */
#define MAX_IJX         20000
    static      char            fl[40];
    FILE                        *fp,*fpg;
    char                        *fname;
    static unsigned char    buffer[SIZEOFBUF],map[SIZEOFBITBUF],buffer2[MAX_BUFF];
    int                         ii1,ii2,jj1,jj2;

    unsigned long               len,mtn;
    unsigned long               len1,iyyyy,imm,iday,ltvn;
    int                         recno;
    unsigned long               len2,ns2;
    unsigned long               len3,ns3,ijmx,Ngdt,Ni,Nj,Di,Dj,mxnp,np[1000];
    long                        La1,La2,Lo1,Lo2;
    unsigned long               len4,ns4,pc,pn,vl1,dfn,kt,lt,tys,Npdt;
    long                        nmem,tyens;
    unsigned long               len5,ns5,ijdim,nbit,ntemplate;
    long                        E,D;
    float                       R;
    unsigned long               len6,ns6,ifbitmap;
    unsigned long               len7,ns7,nb,mb;
    static      float           data[MAX_IJX],wdata[MAX_IJX];
    unsigned long               tcount=0,tcounta=0;
    char                        clvl[5];
/*===========================================================================*/
/* TYPE DEF                                                                  */
/*===========================================================================*/
/*---------------------------------------------------------------------------*/
/* TOOLS                                                                     */
```

```
49  /*-------------------------------------------------------------------------*/
50  void usage(void);
51  long longbybit(unsigned char *buf,unsigned long *pos,unsigned long nbit);
52  long convlong( unsigned char *string, int nbyte );
53  float convfloat( unsigned char *string, int nbyte );
54  long convlatlon(unsigned char *string );
55  char mem2char(char *string);
56
57  /*-------------------------------------------------------------------------*/
58  /*      used in identifydata                                               */
59  /*-------------------------------------------------------------------------*/
60  void fproduct(void);
61  void flevel(void);
62  void fderived(void);
63  void fhemispher(void);
64  void fperiod(void);
65
66  /*-------------------------------------------------------------------------*/
67  /*     Identify Data (FILE NAME)                                           */
68  /*-------------------------------------------------------------------------*/
69  void identifydata(void);
70
71  /*-------------------------------------------------------------------------*/
72  /*     DECODE SECTION                                                      */
73  /*-------------------------------------------------------------------------*/
74  void dcd_sect0(void);
75  void dcd_sect1(void);
76  void dcd_sect2(void);
77  void dcd_sect3(void);
78  void dcd_sect4(void);
79  void dcd_sect5(void);
80  void dcd_sect6(void);
81  void dcd_sect7(void);
82
83  void ini2date(char mmyyyy[]);
84  void str_elem(char celem[]);
85  void out_data(void);
86
87  /*=========================================================================*/
88  /* MAIN PROGRAM                                                            */
89  /*=========================================================================*/
90  main(argc,argv)
91  int       argc;
92  char      **argv;
93  {
94        unsigned long      ns;
95        unsigned long      wkl;
96        char               wk[105],cns[5];
97             /*    Input File    */
98             if(2 != argc)
99                        {
100                       usage();
101                       exit(0);
```

```
102                     }
103             if(NULL == (fp = fopen(argv[1],"rb")))
104                     {
105                     printf("¥nCannot open FILE : %s¥n",argv[1]);
106                     usage();
107                     exit(1);
108                     }
109
110             fname=argv[1];
111             recno=0;
112     /*---------------------------------------------------------------------*/
113     /*     DECODE SECT 0   : Indicator Section                             */
114     /*---------------------------------------------------------------------*/
115     SECT0:
116          dcd_sect0();
117     /*---------------------------------------------------------------------*/
118     /*     DECODE SECT 1   : Identification Section                        */
119     /*---------------------------------------------------------------------*/
120         printf("¥n<<SECTION 1>> : Identification Section ¥n");
121         dcd_sect1();
122     /*---------------------------------------------------------------------*/
123     /*     DECODE SECT 2    : (Local Use Section)                          */
124     /*---------------------------------------------------------------------*/
125         if( fread(buffer,sizeof(char), 5,fp) != 5 ){
126             printf("Read ERR SECT 2 !! File(%s)¥n",fname);
127             exit(72);
128             }
129         len2=convlong(&buffer[0], 4);
130         ns2=convlong(&buffer[4], 1);
131         fseek(fp,-5,1);                 /*   Back Space 5 bytes   */
132             switch(ns2){
133                 case 2:
134                     printf("<<SECTION 2>> : (Local Use Section)¥n");
135                     goto SECT2;
136                 case 3:
137                     printf("¥n<<SECTION 3>> : Grid Definition Section¥n");
138                     goto SECT3;
139                 default:
140                     printf("ERROR in Section 8 !!");
141                     printf(" len ns = %d %d¥n",wkl,buffer[0]);
142                     exit(99);
143             }
144     SECT2:
145         dcd_sect2();
146     /*---------------------------------------------------------------------*/
147     /*     DECODE SECT 3       : Grid Definition Section                   */
148     /*---------------------------------------------------------------------*/
149         printf("¥n<<SECTION 3>> : Grid Definition Section¥n");
150     SECT3:
151         dcd_sect3();
152     /*---------------------------------------------------------------------*/
153     /*     DECODE SECT 4         : Product Definition Section              */
154     /*---------------------------------------------------------------------*/
```

```
155         printf("¥n<<SECTION 4>> : Product Definition Section¥n");
156     SECT4:
157         dcd_sect4();
158     /* Identify Data(elm lvl date period */
159         identifydata();
160     /*-------------------------------------------------------------------------*/
161     /*     DECODE SECT 5         : Data Representation Section                 */
162     /*-------------------------------------------------------------------------*/
163         printf("¥n<<SECTION 5>> : Data Representation Section¥n");
164         dcd_sect5();
165     /*-------------------------------------------------------------------------*/
166     /*     DECODE SECT 6          : Bit-map Section                            */
167     /*-------------------------------------------------------------------------*/
168         printf("¥n<<SECTION 6>> : Bit-map Section¥n");
169         dcd_sect6();
170     /*-------------------------------------------------------------------------*/
171     /*     DECODE SECT 7           : Data Section                              */
172     /*-------------------------------------------------------------------------*/
173         printf("¥n<<SECTION 7>> : Data Section¥n");
174         dcd_sect7();
175         if( IFOUT      == 1 ) out_data();                   /*     OutPut Data    */
176
177     /*-------------------------------------------------------------------------*/
178     /*     DECODE SECT 8           : END or LOOP ?                             */
179     /*-------------------------------------------------------------------------*/
180         if( fread(buffer,sizeof(char), 4,fp) != 4 ){
181             printf("Read ERR !! File(%s)¥n",argv[1]);
182             exit(70);
183         }
184         if(buffer[0] == '7' && buffer[1] == '7' &&    buffer[2] == '7' && buffer[3] == '7'){
185             printf("GRIB END !!");
186             goto SECT0;
187             /*exit(0);*/
188             }
189         else{
190             wkl=convlong(&buffer[0], 4);
191             if( fread(buffer,sizeof(char), 1,fp) != 1 ){
192             printf("Read ERR !! File(%s)¥n",argv[1]);
193             exit(71);
194             }
195             fseek(fp,-5,1);                /*   Back Space 5 bytes   */
196             ns=convlong(&buffer[0], 1);
197             switch(ns){
198                 case 2:
199                     printf("¥n----------------¥n");
200                     printf("Record No=%d ¥n",++recno);
201                     printf("----------------¥n");
202                     printf("<<SECTION 2>> : (Local Use Section)¥n");
203                     goto SECT2;
204                 case 3:
205                     printf("¥n----------------¥n");
206                     printf("Record No=%d ¥n",++recno);
207                     printf("----------------¥n");
```

```
208                     printf("¥n<<SECTION 3>> : Grid Definition Section¥n");
209                     goto SECT3;
210                 case 4:
211                     printf("¥n----------¥n");
212                     printf("Record No=%d ¥n",++recno);
213                     printf("----------¥n");
214                     printf("¥n<<SECTION 4>> : Product Definition Section¥n");
215                     goto SECT4;
216                 default:
217                     printf("ERROR in Section 8 !!");
218                     printf(" len ns = %d %d¥n",wkl,buffer[0]);
219                     exit(99);
220             }
221         }
222     }
223
224     /*=========================================================================*/
225     /*      END MAIN()      */
226     /*=========================================================================*/
227     /*    DEFINE FUNCTIONS                                                     */
228     /*=========================================================================*/
229     /*-------------------------------------------------------------------------*/
230     /*    DECODE SECTION 0                                                     */
231     /*-------------------------------------------------------------------------*/
232     void dcd_sect0(void)
233     {
234     /*    DECODE SECT 0                                                        */
235         /*  Check Header  */
236             if( fread(buffer,sizeof(char), 4, fp) == 0 ) exit(2);
237             while( buffer[0] != 'G' || buffer[1] != 'R' ||
238                    buffer[2] != 'I' || buffer[3] != 'B' ){
239                     printf("Cannot Find GRIB header !!¥n");
240
241                     fclose(fp);exit(99);
242                     }
243             printf("----------¥n");
244             printf("---- GRIB FILE !! -----¥n");
245             printf("----------¥n");
246         fread(buffer,sizeof(char), 12, fp);
247         mtn=convlong(&buffer[2], 1);
248         printf("¥n<<SECTION 0>> : Indicator Section¥n");
249         printf("GRIB Master Table Number : %d¥n",mtn);
250         printf("GRIB Edition Number      : %d¥n",buffer[3]);
251         printf("buffer[4]= %d¥n",convlong(&buffer[4],4));
252         if( convlong(&buffer[4],4)  == 0){
253             len = convlong(&buffer[8], 4);
254             printf("Total length of GRIB     : %d¥n",len);
255             }
256         else
257             {
258             printf("Larg Record !! Not Supported !!¥n¥n");
259             exit(3);
260             }
```

```
261     }
262
263     /*-------------------------------------------------------------------------*/
264     /*     DECODE SECTION 1                                                    */
265     /*-------------------------------------------------------------------------*/
266     void dcd_sect1(void)
267     {
268         unsigned long       ns1,idcenter,icsubc;
269
270         fread(buffer,sizeof(char), 5,fp);
271
272         len1=convlong(&buffer[0], 4);
273         ns1=convlong(&buffer[4], 1);
274         fread(buffer,sizeof(char),len1-5,fp);
275         idcenter=convlong(&buffer[0], 2);
276         icsubc=convlong(&buffer[2],2);
277         ltvn=convlong(&buffer[5],1);
278         iyyyy=convlong(&buffer[7], 2);
279         imm=convlong(&buffer[9], 1);
280         iday=convlong(&buffer[10], 1);
281         printf("Length of Section 1    : %d¥n",len1);
282         printf("Number of Section      : %d¥n",ns1);
283         printf("Identification of centre        : %d¥n",idcenter);
284         printf("Identification of sub-centre : %d¥n",icsubc);
285         printf("GRIB Master Tables Version Number : %d¥n",buffer[4]);
286         printf("GRIB Local Tables Version Number   : %d¥n",buffer[5]);
287         printf("Significance of Reference Time      : %d¥n",buffer[6]);
288         printf("Year                          : %d¥n",iyyyy);
289         printf("Month                          : %d¥n",imm);
290         printf("Day                            : %d¥n",iday);
291         printf("Hour                           : %d¥n",buffer[11]);
292         printf("Minute                         : %d¥n",buffer[12]);
293         printf("Second                         : %d¥n",buffer[13]);
294         printf("Production status of processed data : %d¥n",buffer[14]);
295         printf("Type of processed data                    : %d¥n",buffer[15]);
296
297         printf("------------------------¥n");
298         printf("Record No=%d ¥n",++recno);
299         printf("------------------------¥n");
300     }
301
302     /*-------------------------------------------------------------------------*/
303     /*     DECODE SECTION 2                                                    */
304     /*-------------------------------------------------------------------------*/
305     void dcd_sect2(void)
306     {
307         if( fread(buffer,sizeof(char), 5,fp) != 5 ){
308             printf("Read ERR SECT 2 !! File(%s)¥n",fname);
309             exit(72);
310             }
311         len2=convlong(&buffer[0], 4);
312         ns2=convlong(&buffer[4], 1);
313         printf("Length of Section 2    : %d¥n",len2);
```

```
314         printf("Number of Section      : %d¥n",ns2);
315         if( len2-4 != 0 ){
316             fread(buffer,sizeof(char),len2-5,fp);
317             printf(" Local use .... Not Supported!!");
318             exit(2);
319         }
320     }
321
322     /*---------------------------------------------------------------------------*/
323     /*    DECODE SECTION 3                                                       */
324     /*---------------------------------------------------------------------------*/
325     void dcd_sect3(void)
326     {
327         unsigned long wkl,i,ol,Ntoe;
328
329         if( fread(buffer,sizeof(char), 5,fp) != 5 ){
330             printf("Read ERR SECT 3 !! File(%s)¥n",fname);
331             exit(73);
332             }
333         len3=convlong(&buffer[0], 4);
334         ns3=convlong(&buffer[4], 1);
335
336         fread(buffer,sizeof(char),len3-5,fp);
337         ijmx = convlong(&buffer[1], 4);
338         Ngdt = convlong(&buffer[7], 2);
339         ol     = convlong(&buffer[5], 1);
340         printf("Length of section                    : %d¥n",len3);
341         printf("Number of section                    : %d¥n",ns3);
342         printf("Source of grid definition            : %d¥n",buffer[0]);
343         printf("Number of data points                : %d¥n",ijmx);
344         printf("optional list                        : %d¥n",ol);
345         printf("Interpretation of list               : %d¥n",buffer[6]);
346         printf("Grid Definition Template Number : %d¥n",Ngdt);
347         switch(Ngdt){
348             case      0:
349                 printf("((Grid Definition Template 3.0))¥n");
350                 printf("Shape of the earth                                    : %d¥n",buffer[9]);
351                 printf("Scale factor of radius of spherical earth             : %d¥n",buffer[10]);
352                 wkl = convlong(&buffer[11], 4);
353                 printf("Scaled value of radius of spherical earth             : %u¥n",wkl);
354                 printf("Scale factor of major axis of oblate spheroid earth   : %d¥n",buffer[15]);
355                 wkl =   convlong(&buffer[16], 4);
356                 printf("Scaled value of major axis of oblate spheroid earth   : %u¥n",wkl);
357                 printf("Scale factor of minor axis of oblate spheroid earth   : %d¥n",buffer[20]);
358                 wkl = convlong(&buffer[21], 4);
359                 printf("Scaled value of minor axis of oblate spheroid earth   : %u¥n",wkl);
360                 Ni   = convlong(&buffer[25], 4);
361                 Nj   = convlong(&buffer[29], 4);
362                 printf("number of points along a parallel                     : %d¥n",Ni);
363                 printf("number of points along a meridian                     : %d¥n",Nj);
364                 wkl = convlong(&buffer[33], 4);
365                 printf("Basic angle of the initial production domain          : %d¥n",wkl);
366                 wkl = convlong(&buffer[37], 4);
```

```
367             printf("Subdivisions of basic angle                    : %u¥n",wkl);
368             La1 = convlong(&buffer[41], 4);
369             Lo1 = convlong(&buffer[45], 4);
370             printf("latitude of first grid point(La1)              : %d¥n",La1);
371             printf("longitude of first grid point(Lo1)             : %d¥n",Lo1);
372             printf("Resolution and component flags                 : %d¥n",buffer[49]);
373             La2 = convlatlon(&buffer[50]);
374             Lo2 = convlatlon(&buffer[54]);
375             printf("latitude of last grid point(La2)               : %d¥n",La2);
376             printf("longitude of last grid point(Lo2)              : %d¥n",Lo2);
377             Di  = convlong(&buffer[58], 4);
378             Dj  = convlong(&buffer[62], 4);
379             printf("i direction increment(Di)                      : %d¥n",Di);
380             printf("j direction increment(Dj)                      : %d¥n",Dj);
381             printf("Scanning mode                                  : %d¥n",buffer[66]);
382             break;
383         case 40:  /*  Not Checked  TEST TEST TEST  */
384             printf("((Grid Definition Template 3.40))¥n");
385             printf("Shape of the earth                             : %d¥n",buffer[9]);
386             printf("Scale factor of radius of spherical earth      : %d¥n",buffer[10]);
387             wkl = convlong(&buffer[11], 4);
388             printf("Scaled value of radius of spherical earth      : %d¥n",wkl);
389             printf("Scale factor of major axis of oblate spheroid earth : %d¥n",buffer[15]);
390             wkl =  convlong(&buffer[16], 4);
391             printf("Scaled value of major axis of oblate spheroid earth : %d¥n",wkl);
392             printf("Scale factor of minor axis of oblate spheroid earth : %d¥n",buffer[20]);
393             wkl = convlong(&buffer[21], 4);
394             printf("Scaled value of minor axis of oblate spheroid earth : %d¥n",wkl);
395             Ni  = convlong(&buffer[25], 4);
396             Nj  = convlong(&buffer[29], 4);
397             printf("number of points along a parallel              : %d¥n",Ni);
398             printf("number of points along a meridian              : %d¥n",Nj);
399             wkl = convlong(&buffer[33], 4);
400             printf("Basic angle of the initial production domain   : %d¥n",wkl);
401             wkl = convlong(&buffer[37], 4);
402             printf("Subdivisions of basic angle                    : %d¥n",wkl);
403             La1 = convlong(&buffer[41], 4);
404             Lo1 = convlong(&buffer[45], 4);
405             printf("latitude of first grid point(La1)              : %d¥n",La1);
406             printf("longitude of first grid point(Lo1)             : %d¥n",Lo1);
407             printf("Resolution and component flags                 : %d¥n",buffer[49]);
408             La2 = convlatlon(&buffer[50]);
409             Lo2 = convlatlon(&buffer[54]);
410             printf("latitude of last grid point(La2)               : %d¥n",La2);
411             printf("longitude of last grid point(Lo2)              : %d¥n",Lo2);
412             Di  = convlong(&buffer[58], 4);
413             Ntoe= convlong(&buffer[62], 4);
414             printf("i direction increment(Di)                      : %d¥n",Di);
415             printf("Number of parallels from pole to equator(N)    : %d¥n",Ntoe);
416             printf("Scanning mode                                  : %d¥n",buffer[66]);
417             printf("Number of points along each grid line ¥n");
418             printf("      Line      Points¥n");
419             mxnp=0;
```

```
420                     for(i=0 ;i < Nj ;++i)
421                         {
422                         np[i]=convlong(&buffer[67+i*2], ol );
423                         if(np[i] > mxnp) mxnp=np[i];
424                         }
425                     break;
426                 default      :
427                     printf("This Grid is Not Supported !!");
428                     exit(10);
429             }
430     }
431
432     /*---------------------------------------------------------------------------*/
433     /*     DECODE SECTION 4                                                      */
434     /*---------------------------------------------------------------------------*/
435     void dcd_sect4(void)
436     {
437         unsigned long       noc,tcut,svl1,svl2;
438         unsigned long       iyyyye,imme,idaye,Nt,nmiss,tinc;
439         unsigned long       i,latn,lats,lone,lonw,Nc,vstd,vdist;
440                     char        sfl1;
441
442         if( fread(buffer,sizeof(char), 5,fp) != 5 ){
443             printf("Read ERR SECT 4 !! File(%s)¥n",fname);
444             exit(74);
445             }
446         len4=convlong(&buffer[0], 4);
447         ns4=convlong(&buffer[4], 1);
448
449         fread(buffer,sizeof(char),len4-5,fp);
450         noc = convlong(&buffer[0], 2);
451         Npdt= convlong(&buffer[2], 2);
452         printf("Length of section 4                                    : %d¥n",len4);
453         printf("Number of section                                      : %d¥n",ns4);
454         printf("Number of coordinates values after Template    : %d¥n",noc);
455         printf("Product Definition Template Number                     : %d¥n",Npdt);
456         printf("(( Product Definition Template 4.%d ))¥n",Npdt);
457         switch(Npdt){
458
459             case        1:                          /*     PDT4. 1     */
460                 pc        = convlong(&buffer[4], 1);
461                 pn        = convlong(&buffer[5], 1);
462                 tcut      = convlong(&buffer[9], 2);
463                 kt        = convlong(&buffer[13], 4);
464                 tys       = convlong(&buffer[17], 1);
465                 sfl1      = buffer[18];
466                 sfl1      = mem2char(&sfl1);
467                 svl1      = convlong(&buffer[19], 4);
468                 vl1       = svl1*pow(10.0,-1*(double)sfl1);
469                 svl2      = convlong(&buffer[25], 4);
470                 tyens     = convlong(&buffer[29], 1);
471                 nmem      = convlong(&buffer[30], 1);
472                 /*iyyyye = convlong(&buffer[32], 2);
```

```
473             imme    = convlong(&buffer[34], 1);
474             idaye   = convlong(&buffer[35], 1);
475             Nt      = convlong(&buffer[39], 1);
476             nmiss   = convlong(&buffer[40], 4);*/
477             printf("Parameter category                              : %d¥n",pc);
478             printf("Parameter number                                : %d¥n",pn);
479             printf("Type of generating process                      : %d¥n",buffer[6]);
480             printf("Background generating process identifier        : %d¥n",buffer[7]);
481             printf("Forecast generating process identifier          : %d¥n",buffer[8]);
482             printf("Hours after reference time of data cut-off      : %d¥n",tcut);
483             printf("Minutes after reference time of data cut-off    : %d¥n",buffer[11]);
484             printf("Indicator of unit of time range                 : %d¥n",buffer[12]);
485             printf("Forecast time in units defined by octet 18      : %d¥n",kt);
486             printf("Type of first fixed surface                     : %d¥n",tys);
487             printf("Scale factor of first fixed surface             : %d¥n",sfl1);
488             printf("Scaled value of first fixed surface             : %d¥n",svl1);
489             printf("Type of second fixed surface                    : %d¥n",buffer[23]);
490             printf("Scale factor of second fixed surface            : %d¥n",buffer[24]);
491             printf("Scaled value of second fixed surface            : %d¥n",svl2);
492             printf("Type of ensemble forecast                       : %d¥n",buffer[29]);
493             printf("Perturbation number                             : %d¥n",buffer[30]);
494             printf("Number of forecasts in ensemble                 : %d¥n",buffer[31]);
495             break;
496         case    2:                      /*  PDT4.2  */
497             pc      = convlong(&buffer[4], 1);
498             pn      = convlong(&buffer[5], 1);
499             tcut    = convlong(&buffer[9], 2);
500             kt      = convlong(&buffer[13], 4);
501             tys     = convlong(&buffer[17], 1);
502             sfl1    = buffer[18];
503             sfl1    = mem2char(&sfl1);
504             svl1    = convlong(&buffer[19], 4);
505             vl1     = svl1*pow(10.0,-1*(double)sfl1);
506             svl2    = convlong(&buffer[25], 4);
507             dfn     = convlong(&buffer[29], 1);
508             nmem    = -1;
509             printf("Parameter category                              : %d¥n",pc);
510             printf("Parameter number                                : %d¥n",pn);
511             printf("Type of generating process                      : %d¥n",buffer[6]);
512             printf("Background generating process identifier        : %d¥n",buffer[7]);
513             printf("Forecast generating process identifier          : %d¥n",buffer[8]);
514             printf("Hours after reference time of data cut-off      : %d¥n",tcut);
515             printf("Minutes after reference time of data cut-off    : %d¥n",buffer[11]);
516             printf("Indicator of unit of time range                 : %d¥n",buffer[12]);
517             printf("Forecast time in units defined by octet 18      : %d¥n",kt);
518             printf("Type of first fixed surface                     : %d¥n",tys);
519             printf("Scale factor of first fixed surface             : %d¥n",sfl1);
520             printf("Scaled value of first fixed surface             : %u¥n",svl1);
521             printf("Type of second fixed surface                    : %d¥n",buffer[23]);
522             printf("Scale factor of second fixed surface            : %d¥n",buffer[24]);
523             printf("Scaled value of second fixed surface            : %d¥n",svl2);
524             printf(" Derived forecast (see Code Table 4.7)          : %d¥n",dfn);
525             printf("Number of forecasts in ensemble                 : %d¥n",buffer[30]);
```

```
526                 break;
527             case    11:                             /*    PDT4.11    */
528                 pc      = convlong(&buffer[4], 1);
529                 pn      = convlong(&buffer[5], 1);
530                 tcut    = convlong(&buffer[9], 2);
531                 kt      = convlong(&buffer[13], 4);
532                 tys     = convlong(&buffer[17], 1);
533                 sfl1    = buffer[18];
534                 sfl1    = mem2char(&sfl1);
535                 svl1    = convlong(&buffer[19], 4);
536                 vl1     = svl1*pow(10.0,-1*(double)sfl1);
537                 svl2    = convlong(&buffer[25], 4);
538                 tyens   = convlong(&buffer[29], 1);
539                 nmem    = convlong(&buffer[30], 1);
540                 iyyyye = convlong(&buffer[32], 2);
541                 imme    = convlong(&buffer[34], 1);
542                 idaye   = convlong(&buffer[35], 1);
543                 Nt      = convlong(&buffer[39], 1);
544                 nmiss   = convlong(&buffer[40], 4);
545                 printf("Parameter category                                  : %d¥n",pc);
546                 printf("Parameter number                                    : %d¥n",pn);
547                 printf("Type of generating process                          : %d¥n",buffer[6]);
548                 printf("Background generating process identifier            : %d¥n",buffer[7]);
549                 printf("Forecast generating process identifier              : %d¥n",buffer[8]);
550                 printf("Hours after reference time of data cut-off          : %d¥n",tcut);
551                 printf("Minutes after reference time of data cut-off        : %d¥n",buffer[11]);
552                 printf("Indicator of unit of time range                     : %d¥n",buffer[12]);
553                 printf("Forecast time in units defined by octet 18          : %d¥n",kt);
554                 printf("Type of first fixed surface                         : %d¥n",tys);
555                 printf("Scale factor of first fixed surface                 : %d¥n",sfl1);
556                 printf("Scaled value of first fixed surface                 : %d¥n",svl1);
557                 printf("Type of second fixed surface                        : %d¥n",buffer[23]);
558                 printf("Scale factor of second fixed surface                : %d¥n",buffer[24]);
559                 printf("Scaled value of second fixed surface                : %d¥n",svl2);
560                 printf("Type of ensemble forecast                           : %d¥n",buffer[29]);
561                 printf("Perturbation number                                 : %d¥n",buffer[30]);
562                 printf("Number of forecasts in ensemble                     : %d¥n",buffer[31]);
563                 printf("Year                                : %d¥n",iyyyye);
564                 printf("Month                               : %d¥n",imme);
565                 printf("Day                                 : %d¥n",idaye);
566                 printf("Hour                                : %d¥n",buffer[36]);
567                 printf("Minut                               : %d¥n",buffer[37]);
568                 printf("Second                              : %d¥n",buffer[38]);
569                 printf("n - Number of time range            : %d¥n",Nt);
570                 printf("Total number of data values missing                 : %d¥n",nmiss);
571                 for(i=0 ; i < Nt ; ++i){
572                     lt      = convlong(&buffer[47+i*12], 4);
573                     tinc    = convlong(&buffer[52+i*12], 4);
574                     printf("Statistical process used to calculate ....      : %d¥n",buffer[44+i*12]);
575                     printf("Type of time increment                              : %d¥n",buffer[45+i*12]);
576                     printf("Indicator of unit of time for time range            : %d¥n",buffer[46+i*12]);
577                     printf("Length of the time range                            : %d¥n",lt);
578                     printf("Indicator of unit of time for the increment         : %d¥n",buffer[51+i*12]);
```

```
579                     printf("Time increment between successive fields          : %d¥n",tinc);
580                 }
581                 break;
582             case    12:                     /*    PDT4.12    */
583                 pc      = convlong(&buffer[4], 1);
584                 pn      = convlong(&buffer[5], 1);
585                 tcut    = convlong(&buffer[9], 2);
586                 kt      = convlong(&buffer[13], 4);
587                 tys     = convlong(&buffer[17], 1);
588                 sfl1    = buffer[18];
589                 sfl1    = mem2char(&sfl1);
590                 svl1    = convlong(&buffer[19], 4);
591                 vl1     = svl1*pow(10.0,-1*(double)sfl1);
592                 svl2    = convlong(&buffer[25], 4);
593                 dfn     = convlong(&buffer[29], 1);
594                 nmem    = -1;
595                 iyyyye = convlong(&buffer[31], 2);
596                 imme    = convlong(&buffer[33], 1);
597                 idaye   = convlong(&buffer[34], 1);
598                 Nt      = convlong(&buffer[38], 1);
599                 nmiss   = convlong(&buffer[39], 4);
600                 printf("Parameter category                                : %d¥n",pc);
601                 printf("Parameter number                                  : %d¥n",pn);
602                 printf("Type of generating process                        : %d¥n",buffer[6]);
603                 printf("Background generating process identifier          : %d¥n",buffer[7]);
604                 printf("Forecast generating process identifier            : %d¥n",buffer[8]);
605                 printf("Hours after reference time of data cut-off        : %d¥n",tcut);
606                 printf("Minutes after reference time of data cut-off      : %d¥n",buffer[11]);
607                 printf("Indicator of unit of time range                   : %d¥n",buffer[12]);
608                 printf("Forecast time in units defined by octet 18        : %d¥n",kt);
609                 printf("Type of first fixed surface                       : %u¥n",tys);
610                 printf("Scale factor of first fixed surface               : %u¥n",sfl1);
611                 printf("Scaled value of first fixed surface               : %u¥n",svl1);
612                 printf("Type of second fixed surface                      : %d¥n",buffer[23]);
613                 printf("Scale factor of second fixed surface              : %d¥n",buffer[24]);
614                 printf("Hours after reference time of data cut-off        : %d¥n",tcut);
615                 printf("Minutes after reference time of data cut-off      : %d¥n",buffer[11]);
616                 printf("Indicator of unit of time range                   : %d¥n",buffer[12]);
617                 printf("Forecast time in units defined by octet 18        : %d¥n",kt);
618                 printf("Type of first fixed surface                       : %d¥n",tys);
619                 printf("Scale factor of first fixed surface               : %d¥n",sfl1);
620                 printf("Scaled value of first fixed surface               : %d¥n",svl1);
621                 printf("Type of second fixed surface                      : %d¥n",buffer[23]);
622                 printf("Scale factor of second fixed surface              : %d¥n",buffer[24]);
623                 printf("Scaled value of second fixed surface              : %d¥n",svl2);
624                 printf("Derived forecast (see Code Table 4.7)             : %d¥n",dfn);
625                 printf("Number of forecasts in the ensemble (N)           : %d¥n",buffer[30]);
626                 printf("Cluster identifier                                : %d¥n",buffer[31]);
627                 printf("Number of cluster to which the high resolution control
628     belongs : %d¥n",buffer[32]);
629                 printf("Number of cluster to which the low resolution control
630     belongs   : %d¥n",buffer[33]);
631                 printf("Total number of clusters                          : %d¥n",buffer[34]);
```

```
632                 printf("Clustering method (see Code Table 4.8)              : %d¥n",buffer[35]);
633                 printf("Northern latitude of cluster domain                 : %d¥n",latn);
634                 printf("Southern latitude of cluster domain                 : %d¥n",lats);
635                 printf("Eastern longitude of cluster domain                 : %d¥n",lone);
636                 printf("Western longitude of cluster domain                 : %d¥n",lonw);
637                 printf("Number of forecasts in the cluster                  : %d¥n",Nc);
638                 printf("Scale factor of standard deviation in the cluster           : %d¥n",buffer[53]);
639                 printf("Scaled value of standard deviation in the cluster           : %u¥n",vstd);
640                 printf("Scale factor of distance of the cluster from ensemble mean  : %d¥n",buffer[58]);
641                 printf("Scaled value of distance of the cluster from ensemble mean  : %u¥n",vdist);
642                 printf("Year                                : %d¥n",iyyyye);
643                 printf("Month                               : %d¥n",imme);
644                 printf("Day                                 : %d¥n",idaye);
645                 printf("Hour                                : %d¥n",buffer[67]);
646                 printf("Minut                               : %d¥n",buffer[68]);
647                 printf("Second                              : %d¥n",buffer[69]);
648                 printf("n - Number of time range        : %d¥n",Nt);
649                 printf("Total number of data values missing             : %d¥n",nmiss);
650                 for(i=0 ; i < Nt ; ++i){
651                     lt      = convlong(&buffer[78+i*12], 4);
652                     tinc    = convlong(&buffer[83+i*12], 4);
653                     printf("Statistical process used to calculate ....      : %d¥n",buffer[75+i*12]);
654                     printf("Type of time increment                              : %d¥n",buffer[76+i*12]);
655                     printf("Indicator of unit of time for time range        : %d¥n",buffer[77+i*12]);
656                     printf("Length of the time range                            : %d¥n",lt);
657                     printf("Indicator of unit of time for the increment     : %d¥n",buffer[82+i*12]);
658                     printf("Time increment between successive fields        : %d¥n",tinc);
659                 }
660                     printf("List of Nc ensemble forecast numbers :");
661                 for(i=0 ; i < Nc ; ++i){
662                     printf(" %2d",buffer[74+Nt*12+i+1]);
663                 }
664                     printf("¥n");
665                 break;
666             default:
667                 printf("This Product Definition Template is Not Supported !!");
668                 exit(11);
669         }
670     }
671     /*---------------------------------------------------------------------------*/
672     /*    DECODE SECTION 5                                                       */
673     /*---------------------------------------------------------------------------*/
674     void dcd_sect5(void)
675     {
676         fread(buffer,sizeof(char), 5,fp);
677         len5=convlong(&buffer[0], 4);
678         ns5=convlong(&buffer[4], 1);
679         fread(buffer,sizeof(char),len5-5,fp);
680         ijdim = convlong(&buffer[0], 4);
681         ntemplate = convlong(&buffer[4], 2);
682
683         printf("Length of section in octets                 : %d¥n",len5);
684         printf("Number of section                           : %d¥n",ns5);
```

```
685         printf("Number of data points                          : %d¥n",ijdim);
686         printf("Data Representation Template Number            : %d¥n",ntemplate);
687         switch(ntemplate){
688             case      0:
689                 R = convfloat(&buffer[6], 4);
690                 E = convlong(&buffer[10], 2);
691                     if(E > 32768) E = (E - 32768 )*(-1);
692                 D = convlong(&buffer[12], 2);
693                     if(D > 32768) D = (D - 32768 )*(-1);
694                 nbit = convlong(&buffer[14], 1);
695                 printf("Reference value (R)                          : %f¥n",R);
696                 printf("Binary scale factor (E)                      : %d¥n",E);
697                 printf("Decimal scale factor (D)                     : %d¥n",D);
698                 printf("Number of bits .....                         : %d¥n",nbit);
699                 printf("Type of original field values                : %d¥n",buffer[15]);
700                 break;
701             default:
702                     printf("This Data Representation Template is Not Supported!!");
703                     exit(12);
704             }
705     }
706
707     /*-------------------------------------------------------------------------*/
708     /*    DECODE SECTION 6                                                     */
709     /*-------------------------------------------------------------------------*/
710     void dcd_sect6(void)
711     {
712         int      i;
713
714         fread(buffer,sizeof(char), 5,fp);
715         len6=convlong(&buffer[0], 4);
716         ns6=convlong(&buffer[4], 1);
717         fread(buffer,sizeof(char), 1,fp);
718         ifbitmap = convlong(&buffer[0], 1);
719         printf("Length of section in octets                    : %d¥n",len6);
720         printf("Number of section                                : %d¥n",ns6);
721         printf("Bit-map indicator                                : %d¥n",ifbitmap);
722         for( i = 0 ; i < SIZEOFBITBUF ;++i){
723             map[i]=0xFF;
724             }
725         switch(ifbitmap){
726         case         0:
727             if ( len6-6 != fread(map,sizeof(char),len6-6,fp)){
728                 printf("Read ERR SECT 6-1 !! File(%s)¥n",fname);
729                 exit(100);
730                 }
731             break;
732             case      255:
733                 printf("No BITMAP !!¥n");
734                 break;
735             default:
736                 printf("This Bit-map indicator is Not Supportted!!");
737                 exit(13);
```

```
738         }
739     }
740
741     /*---------------------------------------------------------------------*/
742     /*     DECODE SECTION 7                                                */
743     /*---------------------------------------------------------------------*/
744     void dcd_sect7(void)
745     {
746         unsigned char       msk_bit=MASK1;
747         unsigned long       posi,lgpv,i;
748
749
750         fread(buffer,sizeof(char), 5,fp);
751         len7=convlong(&buffer[0], 4);
752         ns7=convlong(&buffer[4], 1);
753         printf("Length of section in octets                    : %d¥n",len7);
754         printf("Number of section                                 : %d¥n",ns7);
755         fread(buffer2,sizeof(char),len7-5,fp);
756         printf("ntemplate=%d¥n",ntemplate);
757         switch(ntemplate){
758             case 0:
759
760                 posi = 0;
761                 for( i = 0 ;i < ijmx ;i++ ){
762                     nb = i / 8;
763                     mb = i % 8;
764                     if( 0 ==    (map[nb] & ( msk_bit >> mb ))){
765                         data[i]=UNDEF;
766                         }
767                     else{
768                         lgpv=longbybit(buffer2,&posi,nbit);
769                         data[i] = (R + lgpv*pow(2.0,(double)E))/pow(10.0,(double)D);
770                         }
771                     }
772
773                 printf("Check DATA !!       ¥n");
774                 printf("     No        value ¥n");
775                 for( i = 0; i < 10 ; ++i){
776                     printf("    %5d       %f¥n",i+1,data[i]);
777                 }
778                 printf("¥n¥n");
779                 for( i = ijmx-10; i < ijmx ; ++i){
780                     printf("    %5d       %f¥n",i+1,data[i]);
781                 }
782
783                 break;
784             default:
785                 printf("Unsupported Template !!");
786                 exit(14);
787         }
788     }
789
790     /*---------------------------------------------------------------------*/
```

```
/*    OUT_DATA                                                                  */
/*--------------------------------------------------------------------------*/
void out_data(void)
{
    int                 itot;
    unsigned int        i,j,ii,flen;
    FILE                *fpc1;
    char                flc1[31];
    char                celem[100],mmyyyy[7];
    float               fla1,fdi,flo1,fdj;

    if(( fpg = fopen(fl,"wb")) == NULL ){
        printf("Stop, Create File(%s)¥n",fl);
        exit(99);
        }

    switch(Ngdt)
        {
        case 0:
            if( fwrite(data,4,ijmx,fpg) == ijmx )
            {
                /*    Out Put GrADS Control File   */
                strcpy(flc1,fl);
                flen=strlen(flc1);
                flc1[flen-4      ]='.';flc1[flen - 3]='c';flc1[flen - 2]='t';
                flc1[flen - 1]='l';flc1[flen]=0x00;
                if(( fpc1 = fopen(flc1,"wt")) == NULL ){
                    printf("Stop, Create File(%s)¥n",flc1);
                    exit(98);
                    }
                fla1=(-1)*(float)La1/1000000.0 ;fdi=(float)Di/1000000.0;
                flo1=(float)Lo1/1000000.0 ;fdj=(float)Dj/1000000.0;

                ini2date(mmyyyy);
                str_elem(celem);
                tcount=1;

                fprintf(fpc1,"dset ^%s¥n",fl);
                fprintf(fpc1,"undef -19999.¥n");
                fprintf(fpc1,"xdef    %d    linear    %f   %f¥n",Ni,flo1,fdi);
                fprintf(fpc1,"ydef    %d     linear   %f   %f¥n",Nj,fla1,fdj);
                fprintf(fpc1,"zdef    1      linear   %s   1 ¥n",clvl);
                fprintf(fpc1,"tdef    %d      linear    01%s   1mo¥n",tcount,mmyyyy);
                fprintf(fpc1,"vars    1 ¥n");
                fprintf(fpc1,"%s¥n",celem);
                fprintf(fpc1,"endvars¥n");
                fprintf(fpc1,"options yrev¥n");
                printf("Close Control File= %s ¥n",flc1);
                if ( fclose(fpc1) != 0 ){
                    printf("Close ERR !! File(%s)¥n",flc1);
                    exit(96);
                    }
                }
```

```
844                 else{
845                     printf("Write ERR !! File(%s)¥n",fl);
846                     exit(98);
847                     }
848                 break;
849             default:
850                 printf(" This GDT is not supported !!   Ngdt= %d ¥n",Ngdt);
851                 exit(95);
852             }
853         printf("Close GrADS File= %s ¥n",fl);
854         if ( fclose(fpg) != 0 ){
855             printf("Close ERR !! File(%s)¥n",fl);
856             exit(97);
857             }
858     }
859
860     /*=========================================================================*/
861     void ini2date(char mmyyyy[])
862     {
863         unsigned long yyyy1,mm1;
864         char      yyyy[5],mm[3];
865
866         if       ( imm == 12){ strcpy(mm,"Jan");yyyy1=iyyyy+1;}
867         else if( imm ==   1){ strcpy(mm,"Feb");yyyy1=iyyyy   ;}
868         else if( imm ==   2){ strcpy(mm,"Mar");yyyy1=iyyyy   ;}
869         else if( imm ==   3){ strcpy(mm,"Apr");yyyy1=iyyyy   ;}
870         else if( imm ==   4){ strcpy(mm,"May");yyyy1=iyyyy   ;}
871         else if( imm ==   5){ strcpy(mm,"Jun");yyyy1=iyyyy   ;}
872         else if( imm ==   6){ strcpy(mm,"Jul");yyyy1=iyyyy   ;}
873         else if( imm ==   7){ strcpy(mm,"Aug");yyyy1=iyyyy   ;}
874         else if( imm ==   8){ strcpy(mm,"Sep");yyyy1=iyyyy   ;}
875         else if( imm ==   9){ strcpy(mm,"Oct");yyyy1=iyyyy   ;}
876         else if( imm == 10){ strcpy(mm,"Nov");yyyy1=iyyyy   ;}
877         else if( imm == 11){ strcpy(mm,"Dec");yyyy1=iyyyy   ;}
878         else{ printf(" Initial month is strange!! imm=%d¥n",imm); exit(55);}
879
880         sprintf(mmyyyy,"%s%04d",mm,yyyy1);
881     }
882
883     void str_elem(char celem[])
884     {
885         char              elm[5],lvl[5],spr[5],comm[70],wstr[10];
886         char              sprcomm[20];
887         unsigned long vl1hpa;
888         unsigned int   i,len1,len2,len3,len4,len5,len6;
889
890         sprcomm[0]=0x00;
891         elm[0]=0x00;lvl[0]=0x00;spr[0]=0x00;
892         vl1hpa=vl1*0.01;
893
894      /*--------------- LEVEL     ------------------------------------------*/
895         if       ( tys == 1     ){strcpy(lvl,"surf")            ;strcpy(clvl,"1000");}
896         else if( tys == 101 ){strcpy(lvl,"sea")             ;strcpy(clvl,"1000");}
```

```
897         else if( tys == 103 ){strcpy(lvl,"2m")              ;strcpy(clvl,"1000");}
898         else if( tys == 100 ){sprintf(lvl,"%01d",vl1hpa);sprintf(clvl,"%3d",vl1hpa);}
899         else     {printf(" This level is Not Supported !! tys %d¥n",tys);
900                   exit(43);}
901       /*--------------- SPREAD ? -------------------------------------*/
902         if ( dfn == 4 ){
903           strcpy(spr,"s");strcpy(sprcomm,"spread ");
904           }
905         else if( dfn == 5 ){
906           strcpy(spr,"l");strcpy(sprcomm,"larg anomaly index ");
907           }
908
909       /*--------------- ELEMENT ---------------------------------------*/
910       if( mtn == 0)
911           {
912           if       ( pc == 0 && pn == 0 ){
913             if( tys != 100 ){
914               strcpy(elm,"t")     ;sprintf(comm,"Temperature %s at %s    [K]",sprcomm,lvl);}
915             else{
916               strcpy(elm,"t")     ;sprintf(comm,"Temperature at %s hPa [K]",lvl);}
917             }
918           else if( pc == 0 && pn == 9 ){
919             if( tys != 100 ){
920               strcpy(elm,"t")   ;strcpy(spr,"a");
921               sprintf(comm,"Temprature anoaly at %s hPa [K]",lvl);}
922             else{
923               strcpy(elm,"t")   ;strcpy(spr,"a");
924               sprintf(comm,"Temprature anoaly at %s [K]",lvl);}
925             }
926           else if( pc == 1 && pn == 0 ){
927             strcpy(elm,"q")     ;
928             sprintf(comm,"Specific humidity at %s hPa [kg/kg]",lvl);}
929           else if( pc == 1 && pn == 1 ){
930             strcpy(elm,"rh")   ;
931             sprintf(comm,"Relative humidity %s at %s hPa [%%]",sprcomm,lvl);}
932           else if( pc == 1 && pn == 8 ){
933             strcpy(elm,"rr")   ;
934             sprintf(comm,"Total precipitaion %s [mm]",sprcomm);}
935           else if( pc == 1 && pn ==210){
936             strcpy(elm,"rrd")   ;lvl[0]=0x00;
937             sprintf(comm,"Daily mean precipitaion %s [mm]",sprcomm);}
938           else if( pc == 1 && pn ==211){
939             strcpy(elm,"rrd") ;strcpy(spr,"a");lvl[0]=0x00;
940             sprintf(comm,"Daily mean precipitaion anomaly [mm]");}
941           else if( pc == 1 && pn ==212){
942             strcpy(elm,"q")   ;strcpy(spr,"a");
943             sprintf(comm,"Specific humidity anomaly at %s hPa [kg/kg]",lvl);}
944           else if( pc == 1 && pn ==213){
945             strcpy(elm,"rh") ;strcpy(spr,"a");
946             sprintf(comm,"Relative humidity anomaly at %s hPa [%%]",lvl);}
947           else if( pc == 2 && pn == 2 ){
948             strcpy(elm,"u")     ;
949             sprintf(comm,"u wind %s at %s hPa [m/s]",sprcomm,lvl);}
```

```
950           else if( pc == 2 && pn == 3 ){
951             strcpy(elm,"v")     ;
952             sprintf(comm,"v wind %s at %s hPa [m/s]",sprcomm,lvl);}
953           else if( pc == 2 && pn == 4 ){
954             strcpy(elm,"psi") ;
955             sprintf(comm,"Stream function at %s hPa [m^2/s]",lvl);}
956           else if( pc == 2 && pn == 5 ){
957             strcpy(elm,"chi") ;
958             sprintf(comm,"Velocity potential at %s hPa [m^2/s]",lvl);}
959           else if( pc == 2 && pn ==210){
960             strcpy(elm,"u")   ;strcpy(spr,"a");
961             sprintf(comm,"u wind anomaly at %s hPa [m/s]",lvl);}
962           else if( pc == 2 && pn ==211){
963             strcpy(elm,"v")   ;strcpy(spr,"a");
964             sprintf(comm,"v wind anomaly at %s hPa [m/s]",lvl);}
965           else if( pc == 3 && pn == 0 ){
966             if(tys != 100 ){
967               strcpy(elm,"p")     ;
968               sprintf(comm,"Pressure %s at %s    [Pa]",sprcomm,lvl);}
969             else{
970               strcpy(elm,"p")     ;
971               sprintf(comm,"Pressure %s at %s hPa [Pa]",sprcomm,lvl);}
972             }
973           else if( pc == 3 && pn == 1 ){
974             strcpy(elm,"p");
975             sprintf(comm,"Sea level pressure %s [Pa]",sprcomm);}
976           else if( pc == 3 && pn == 5 ){
977             strcpy(elm,"z")     ;
978             sprintf(comm,"Geopotential height %s at %s hPa [gpm]",sprcomm,lvl);}
979           else if( pc == 3 && pn == 8 ){
980             if( tys != 100 ){
981               strcpy(elm,"p")   ; strcpy(spr,"a");
982               sprintf(comm,"Pressure anomaly at %s [Pa]",lvl);}
983             else{
984               strcpy(elm,"p")   ; strcpy(spr,"a");
985               sprintf(comm,"Pressure anomaly at %s hPa [Pa]",lvl);}
986             }
987           else if( pc == 3 && pn == 9 ){
988             strcpy(elm,"z")   ;strcpy(spr,"a");
989             sprintf(comm,"Geopotential height anomaly at %s hPa [gpm]",lvl);}
990           else     {
991             printf(" This element Not Supported !! mtn,pc,pn %d,%d,%d¥n",mtn,pc,pn); exit(40);}
992           }
993         else if(mtn == 10 )
994           {
995           if        ( pc == 3 && pn == 0 ){
996             strcpy(elm,"sst") ;lvl[0]=0x00;
997             sprintf(comm,"Sea surface temperature [K]");}
998           else if( pc == 3 && pn ==192){
999             strcpy(elm,"sst");strcpy(spr,"a");lvl[0]=0x00;
1000            sprintf(comm,"Sea surface temperature anomaly [K]");}
1001          else     {
1002            printf(" This element Not Supported !! mtn,pc,pn %d,%d,%d¥n",mtn,pc,pn); exit(41);}
```

```
1003            }
1004          else
1005            {
1006              printf(" This element Not Supported !! mtn,pc,pn %d,%d,%d¥n",mtn,pc,pn); exit(42);
1007              }
1008       /*--------------- length of string & copy string -----------------------*/
1009          strcpy(wstr,"1 99 ** ");
1010
1011          len1=strlen(elm);
1012          len2=len1 + strlen(lvl);
1013          len3=len2 + strlen(spr);
1014          len4=8 ;
1015          len5=len4 + strlen(wstr);
1016          len6=len5 + strlen(comm);
1017
1018          for (i = 0     ; i < len1 ;++i)celem[i]=elm[i];
1019          for (i = len1; i < len2 ;++i)celem[i]=lvl[i-len1];
1020          for (i = len2; i < len3 ;++i)celem[i]=spr[i-len2];
1021          for (i = len3; i < len4 ;++i)celem[i]=' ';
1022          for (i = len4; i < len5 ;++i)celem[i]=wstr[i-len4];
1023          for (i = len5; i < len6 ;++i)celem[i]=comm[i-len5];
1024          celem[len6]=0x00;
1025
1026       /** printf("len1,len2,len3,len4,len5,len6=%d,%d,%d,%d,%d,%d¥n",len1,len2,len3,len4,len5,len6);
1027          printf(" elm = %s¥n",elm);
1028          printf(" lvl = %s¥n",lvl);
1029          printf(" apr = %s¥n",spr);
1030          printf(" wstr = %s¥n",wstr);
1031          printf(" comm = %s¥n",comm);
1032          printf(" ELEM= %s¥n",celem);
1033       **/
1034       }
1035
1036       /*=========================================================================*/
1037       /*      TOOLS                                                              */
1038       /*=========================================================================*/
1039
1040       void usage(void)
1041       {
1042                 puts("¥n");
1043                 puts("Usage : GRIB2 <FILE-NAME>");
1044       }
1045
1046       long longbybit(unsigned char *buf,unsigned long *pos,unsigned long nbit)
1047       {
1048          char      i,ii;
1049          long      pos8,mg8,rem8,nbytes,remnbit;
1050          long                lout=0;
1051          long                lbig;
1052          unsigned char      *buf2,*cbig;
1053          unsigned char      mask,maskremn,full=0xff;
1054
1055          lbig=1;
```

```
1056        cbig = (unsigned char *)&lbig;
1057
1058        buf2 = (unsigned char *)&lout;
1059        mg8   = *pos % 8;
1060        remnbit= nbit % 8;
1061
1062        if( (*pos + nbit) % 8 == 0 ) rem8 = 0;
1063        else                                rem8 = 8 - (( *pos + nbit ) % 8 );
1064
1065        pos8 = ( *pos + nbit + rem8 ) / 8 - 1;
1066
1067        nbytes = ( nbit ) / 8;
1068        mask    = full >>   mg8 + rem8;
1069        for( i = 0;i < nbytes   ;i++ ){
1070             if( 1 == cbig[3] & 0x01 ) ii = 3 -i;
1071             else                              ii = i    ;
1072             buf2[ii] = ( buf[pos8 - i] >> rem8
1073                          | buf[pos8   - i - 1] << 8 - rem8 );
1074        }
1075
1076        if( 1 == cbig[3] & 0x01 ) ii = 3 - nbytes;
1077        else                              ii = nbytes ;
1078        if( remnbit <= 8 - rem8 ){
1079          maskremn = full >> 8 - remnbit;
1080          buf2[ii] = ( buf[pos8 - nbytes] >> rem8 & maskremn );
1081          }
1082        else{
1083          maskremn = full >> 8 - remnbit & full << 8 - rem8 ;
1084          buf2[ii] = ( buf[pos8 - nbytes] >> rem8
1085                       | buf[pos8 - nbytes -1] << 8 - rem8 & maskremn );
1086          }
1087        *pos = *pos + nbit;
1088        return(lout);
1089    }
1090
1091    long convlong( unsigned char *string, int nbyte )
1092    {
1093        int                 i,ii;
1094        long                numb=0;
1095        long                lbig;
1096        unsigned char    *chrwrk,*cbig;
1097
1098        lbig=1;
1099        cbig = (unsigned char *)&lbig;
1100        /*if( 1 == cbig[3] & 0x01 ) printf("BIG ENDIAN !!");*/
1101        chrwrk = (unsigned char *)&numb;
1102        for( i = 0; i < nbyte; i++ ) {
1103             if( 1 == cbig[3] & 0x01 ) ii = 4 - nbyte + i;
1104             else                              ii = nbyte - 1 - i;
1105
1106             chrwrk[ii] = string[i];
1107        }
1108        return( numb );
```

```
1109    }
1110
1111    float convfloat( unsigned char *string, int nbyte )
1112    {
1113        int                 i,ii;
1114        float               numb=0.;
1115        long                lbig;
1116        unsigned char       *chrwrk,*cbig;
1117
1118        lbig=1;
1119        cbig = (unsigned char *)&lbig;
1120
1121        chrwrk = (unsigned char *)&numb;
1122        for( i = 0; i < nbyte; i++ ) {
1123            if( 1 == cbig[3] & 0x01 ) ii = 4 - nbyte + i;
1124            else                      ii = nbyte - 1 - i;
1125
1126            chrwrk[ii] = string[i];
1127        }
1128        return( numb );
1129    }
1130
1131    long convlatlon(unsigned char *string )
1132    {
1133        long      ll;
1134        char      c,a;
1135        unsigned char wk[5];
1136
1137        c = MASK1 & string[0];
1138        if( 0 != c ){
1139                a = -1;}
1140            else{
1141                a = 1;
1142            }
1143
1144        wk[0] = MASK7 & string[0];
1145        wk[1]=string[1];wk[2]=string[2];wk[3]=string[3];
1146        ll = (long)a*convlong(wk,(int)4);
1147        return(ll);
1148    }
1149
1150    char mem2char(char *string)
1151    {
1152            int rt;
1153            rt = string[0];
1154            if(( rt & MASK1 ) && rt != -1 ) rt = -(rt & MASK7);
1155            return rt;
1156    }
1157
1158    /*=========================================================================*/
1159    /*      used in identifydata                                               */
1160    /*=========================================================================*/
1161    void fproduct(void)
```

```
1162        {
1163        char      wk[4];
1164        fl[0]='X';
1165        fl[1]='X';
1166        fl[2]='X';
1167        if( mtn == 0)
1168          {
1169            if( pc == 0 && pn == 0 ){fl[0]='T';fl[1]='_';fl[2]='_';}
1170          if( pc == 0 && pn == 49){fl[0]='P';fl[1]='W';fl[2]='G';}
1171          if( pc == 0 && pn == 9 ){fl[0]='T';fl[1]='A';fl[2]='_';}
1172          if( pc == 1 && pn == 0 ){fl[0]='Q';fl[1]='Q';fl[2]='_';}
1173          if( pc == 1 && pn == 1 ){fl[0]='R';fl[1]='H';fl[2]='_';}
1174          if( pc == 1 && pn == 8 ){fl[0]='R';fl[1]='R';fl[2]='R';}
1175          if( pc == 1 && pn ==20 ){fl[0]='r';fl[1]='r';fl[2]='r';}
1176          if( pc == 1 && pn ==210){fl[0]='R';fl[1]='R';fl[2]='d';}
1177          if( pc == 1 && pn ==211){fl[0]='R';fl[1]='A';fl[2]='d';}
1178          if( pc == 1 && pn ==212){fl[0]='Q';fl[1]='Q';fl[2]='A';}
1179          if( pc == 1 && pn ==213){fl[0]='R';fl[1]='H';fl[2]='A';}
1180          if( pc == 2 && pn == 1 ){fl[0]='W';fl[1]='_';fl[2]='_';}
1181          if( pc == 2 && pn == 2 ){fl[0]='U';fl[1]='_';fl[2]='_';}
1182          if( pc == 2 && pn == 3 ){fl[0]='V';fl[1]='_';fl[2]='_';}
1183          if( pc == 2 && pn == 4 ){fl[0]='P';fl[1]='S';fl[2]='I';}
1184          if( pc == 2 && pn == 5 ){fl[0]='C';fl[1]='H';fl[2]='I';}
1185          if( pc == 2 && pn ==210){fl[0]='U';fl[1]='A';fl[2]='_';}
1186          if( pc == 2 && pn ==211){fl[0]='V';fl[1]='A';fl[2]='_';}
1187          if( pc == 3 && pn == 0 ){fl[0]='P';fl[1]='_';fl[2]='_';}
1188          if( pc == 3 && pn == 1 ){fl[0]='P';fl[1]='s';fl[2]='a';}
1189          if( pc == 3 && pn == 4 ){fl[0]='z';fl[1]='_';fl[2]='_';}
1190          if( pc == 3 && pn == 5 ){fl[0]='Z';fl[1]='_';fl[2]='_';}
1191          if( pc == 3 && pn == 8 ){fl[0]='P';fl[1]='A';fl[2]='_';}
1192          if( pc == 3 && pn == 9 ){fl[0]='Z';fl[1]='A';fl[2]='_';}
1193          }
1194        else if(mtn == 10 )
1195          {
1196          if( pc == 3 && pn == 0 ){fl[0]='S';fl[1]='T';fl[2]='_';}
1197          if( pc == 3 && pn ==192){fl[0]='S';fl[1]='T';fl[2]='A';}
1198          }
1199        fl[3]='?';
1200        fl[4]='?';
1201        fl[5]='?';
1202        if( nmem <     0 ){fl[3]='_';fl[4]='_';fl[5]='_';}
1203        if( nmem >=    0 && nmem < 100 ){
1204            switch (tyens)
1205              {
1206              case      1:
1207                fl[5] = '_'; break;
1208              case      2:
1209                fl[5] = 'm'; break;
1210              case      3:
1211                fl[5] = 'p'; break;
1212              default:
1213                printf(" tyens Not Supported !! tyens=%d ¥n",tyens);exit(71);
1214              }
```

```
1215              sprintf(wk,"%02d",nmem);fl[3]=wk[0];fl[4]=wk[1];
1216            }
1217          if( nmem >= 100 ){printf(" Over 100 members!! Not Supported.¥n");exit(70);}
1218
1219          }
1220
1221      void flevel(void)
1222          {
1223          char      wk[10];
1224          unsigned long vl1hpa;
1225          vl1hpa=vl1*0.01;
1226          fl[6]='X';
1227          fl[7]='X';
1228          fl[8]='X';
1229          fl[9]='X';
1230          if( tys == 1 ){fl[6]='_';fl[7]='S';fl[8]='F';fl[9]='C';}
1231          if( tys == 100 ){sprintf(wk,"%04d",vl1hpa);
1232                   fl[6]=wk[0];fl[7]=wk[1];fl[8]=wk[2];fl[9]=wk[3];}
1233          if( tys == 101 ){fl[6]='_';fl[7]='S';fl[8]='E';fl[9]='A';}
1234          if( tys == 103 ){fl[6]='_';fl[7]='2';fl[8]='M';fl[9]='_';}
1235
1236          }
1237      void fderived(void)
1238          {
1239          fl[10]='X';fl[11]='X';fl[12]='X';
1240          if( dfn == 0 ){fl[10]='E';fl[11]='S';fl[12]='_';}/* Unweighted mean of all members             */
1241          if( dfn == 1 ){fl[10]='E';fl[11]='S';fl[12]='W';}/* Weighted mean of all members               */
1242          if( dfn == 4 ){fl[10]='S';fl[11]='P';fl[12]='R';}/* Spread of all members                      */
1243          if( dfn == 5 ){fl[10]='L';fl[11]='A';fl[12]='I';}/* Large Anomaly Index of all members         */
1244          if( dfn == 6 ){fl[10]='C';fl[11]='L';fl[12]='S';}/* Unweighted mean of the cluster members */
1245
1246          if( Npdt == 11 ){fl[10]='M';fl[11]='E';fl[12]='M';}/* All member */
1247          }
1248      void fhemispher(void)
1249          {
1250          fl[13]='?';fl[14]='?';
1251          if(La1 >= 80000000 && La2 <= -80000000 ){fl[13]='-';fl[14]='G';}
1252          if(La1 == 90000000 && La2 == 0 ) {fl[13]='-';fl[14]='N';}
1253          if(La1 == 0            && La2 == -90000000 ){fl[13]='-';fl[14]='S';}
1254          if(La1 == 90000000 && La2 == -90000000 ){fl[13]='-';fl[14]='G';}
1255
1256          }
1257      void fperiod(void)
1258          {
1259          fl[15]='.';fl[16]='F';fl[17]='?';fl[18]='?';fl[19]='?';fl[20]='E';
1260          fl[21]='?';fl[22]='?';fl[23]='?';fl[24]='D';fl[25]='?';fl[26]='?';
1261          sprintf(&fl[15],".F%03dM%03dD%02d",kt,imm,iday);
1262
1263          }
1264
1265      /*=============================================================================*/
1266      /*     Identify Data (FILE NAME)                                               */
1267      /*=============================================================================*/
```

```
1268    void identifydata(void)
1269        {
1270        int       i;
1271        for( i = 0 ; i < 31 ; ++i){
1272              fl[i]=' ';
1273              }
1274        fproduct();
1275        flevel();
1276        fderived();
1277        fhemispher();
1278        fperiod();
1279        fl[27]='.';fl[28]='d';fl[29]='a';fl[30]='t';fl[31]=0x00;
1280        printf(" FNAME= %s¥n¥n",fl);
1281        }
1282
```

1か月予報アンサンブル統計全球格子点値ファイルの解読（デコード）処理について

１．解読サンプルプログラムのソースコード
別紙参照。

２．利用方法
以下、解読サンプルプログラムの
ソースコードのファイル名：dcd_1megrib2en_sample.c
実行ファイル名　　：dcd_1merib2en
です。

（１）ccコマンドによりコンパイルしてください。その際標準算術関数を利用可能なようにライブラリをリンクしてください。

実行例）　$ cc dcd_1megrib2en_sample.c – lm – o dcd_1merib2en
(dcd_1megrib2en と言う実行ファイルが生成されます)

- ANSI 準拠の c コンパイラでコンパイルできます。UNIX（HP-UX）及び Linux（RedHat）での動作を確認しています。
- リトルエンディアンマシンにも対応しています。

（２）次のコマンドを入力することにより、１か月予報アンサンブル統計格子点値、各節の内容が端末に表示されると共に、４バイト実数形式でデコードされたデータがファイルに書き出されます。

実行例）$ dcd_1megrib2en ｛1か月予報アンサンブル統計格子点値ファイル名}

- デコードされたデータは、予報時間ごと、要素ごとに、サンプルプログラムで割り付けた複数のファイルに出力されます。ファイル名は247行で使用されている関数identifydata 1461～1474行で割り付けています。
- ファイルの出力を止めたい場合は、サンプルプログラム82行目の
  #define IFOUT　1
  を
  #define IFOUT　0
  など、1以外の数字に変更してください。

別紙

解読サンプルプログラムのソースコード
（１か月予報アンサンブル統計全球格子点値ファイル用）

```
/*****************************************************************************
**   Sample Program to convert   1 month ensemble forecast GPV(GRIB2) of JMA
**                                                  2008.07     JMA/CPD
**
**              Tested on Windows (gcc ,borlandc), Linux (gcc)
**                        and HP-UX (native c compiler)
**
*****************************************************************************/
/*     Description of grobal parameter
*     fl[31]                  :output filename used in out_data()
*     flg[31]                 :output file name used in out_grads() GrADS formatted file
*     flc[31]                 :output file name used in out_grads() GrADS control file
*     *fp,*fpg,*fpc,*fpa   : file pointer corresponding to fl,flg,flc,fla
*     *fname                    : pointer to input file name
*     buffer[SIZEOFBUF]     :buffer to read grib parameter
*     map[SIZEOFBITBUF]     :buffer to read bitmap
*     buffer2[MAX_BUFF]     :buffer to read GPV data
*     recno                   :indicator of number of data
*     < section 0 >
*     len                     :Total length of GRIB
*     mtn                      :GRIB Master Table Number
*     < section 1 >
*     len1                    :Length of Section 1
*     iyyyy,imm,iday          :Reference Time
*     ltvn                    :GRIB Local Tables Version Number
*     < section 2>
*     len2                    :Length of Section 2
*     ns2                     :Number of Section (2 is assumed)
*     < section 3 >
*     len3                    :Length of section 3
*     ns3                     :Number of section (3 is assumed)
*     ijmx                    :Number of data points
*     Ngdt                     :Grid Definition Template Number
*     Ni                      :number of points along a parallel
*     Nj                      :number of points along a meridian
*     Di                      :i direction increment
*     Dj                      :j direction increment
*     La1                     :latitude of first grid point
*     La2                     :latitude of last grid point
*     Lo1                     :longitude of first grid point
*     Lo2                     :longitude of last grid point
*     < section 4 >
*     len4                    :Length of section 4
*     ns4                     :Number of section (4 is assumed)
*     pc                      :Parameter category
*     pn                      :Parameter number
*     vl1                     :value of first fixed surface
*     dfn                     :Derived forecast (see Code Table 4.7)
```

```
 49    *      kt                    :Forecast time in units defined by octet 18
 50    *      lt                    :Length of the time range
 51    *      tys                   :Type of first fixed surface
 52    *      Npdt                   :Product Definition Template Number
 53    *      nmem                    :Perturbation number
 54    *      tyens                 :Type of ensemble forecast
 55    *      < section 5 >
 56    *      len5                  :Length of section 5
 57    *      ns5                    :Number of section
 58    *      ijdim                 :Number of data points
 59    *      nbit                  :Number of bits .....
 60    *      ntemplate              :Data Representation Template Number
 61    *      E                      :Binary scale factor
 62    *      D                      :Decimal scale factor
 63    *      R                      :Reference value
 64    *      < section 6 >
 65    *      len6                  :Length of section 6
 66    *      ns6                    :Number of section (6 is assumed)
 67    *      ifbitmap              :Bit-map indicator
 68    *      < section 7 >
 69    *      len7                  :Length of section 7
 70    *      ns7                    :Number of section (7 is assumed)
 71    *      nb                     :parameter of byte number
 72    *      mb                      :parameter of bit in octet
 73    *      data[MAX_IJX]          :Grid Point Value
 74    *      tcount,tcounta        :counter used in out_grads()
 75    ********************************************************************************/
 76    #include <stdio.h>
 77    #include <string.h>
 78    #include <math.h>
 79    #include <stdlib.h>
 80    #include <ctype.h>
 81
 82    #define IFOUT           1          /*   1 :Output decoded data                */
 83    #define OUTGRADS        0          /*   1 :Output GrADS Formatted Files     */
 84    #define OUTASCII        0          /*   1 :Output Ascii File                  */
 85    #define MASK7           0x7F
 86    #define MASK1           0x80
 87    #define UNDEF          -19999.
 88    #define SIZEOFBUF       10000
 89    #define SIZEOFBITBUF 50000
 90    #define MAX_BUFF        50000      /* buffer size for gpv MAX_BUFF < 4294967295   2^32 */
 91    #define MAX_IJX         20000
 92        static     char            fl[31];
 93        char                       flg[31],flc[31],fla[31];
 94        FILE                        *fp,*fpg,*fpc,*fpa;
 95        char                       *fname;
 96        static unsigned char     buffer[SIZEOFBUF],map[SIZEOFBITBUF],buffer2[MAX_BUFF];
 97        int                        ii1,ii2,jj1,jj2;
 98
 99        unsigned long              len,mtn;
100        unsigned long              len1,iyyyy,imm,iday,ltvn;
101        int                        recno;
```

```
102         unsigned long           len2,ns2;
103         unsigned long           len3,ns3,ijmx,Ngdt,Ni,Nj,Di,Dj,mxnp,np[1000];
104         long                    La1,La2,Lo1,Lo2;
105         unsigned long           len4,ns4,pc,pn,vl1,dfn,kt,lt,tys,Npdt;
106         long                    nmem,tyens;
107         unsigned long           len5,ns5,ijdim,nbit,ntemplate;
108         long                    E,D;
109         float                   R;
110         unsigned long           len6,ns6,ifbitmap;
111         unsigned long           len7,ns7,nb,mb;
112         static    float         data[MAX_IJX],wdata[MAX_IJX];
113         unsigned long           tcount=0,tcounta=0;
114         char                    clvl[5];
115     /*=========================================================================*/
116     /* TYPE DEF                                                                */
117     /*=========================================================================*/
118     /*-------------------------------------------------------------------------*/
119     /* TOOLS                                                                   */
120     /*-------------------------------------------------------------------------*/
121     void usage(void);
122     long longbybit(unsigned char *buf,unsigned long *pos,unsigned long nbit);
123     long convlong( unsigned char *string, int nbyte );
124     float convfloat( unsigned char *string, int nbyte );
125     long convlatlon(unsigned char *string );
126     char mem2char(char *string);
127
128     /*-------------------------------------------------------------------------*/
129     /*      used in identifydata                                               */
130     /*-------------------------------------------------------------------------*/
131     void fproduct(void);
132     void flevel(void);
133     void fderived(void);
134     void fhemispher(void);
135     void fperiod(void);
136
137     /*-------------------------------------------------------------------------*/
138     /*    Identify Data (FILE NAME)                                            */
139     /*-------------------------------------------------------------------------*/
140     void identifydata(void);
141
142     /*-------------------------------------------------------------------------*/
143     /*    DECODE SECTION                                                       */
144     /*-------------------------------------------------------------------------*/
145     void dcd_sect0(void);
146     void dcd_sect1(void);
147     void dcd_sect2(void);
148     void dcd_sect3(void);
149     void dcd_sect4(void);
150     void dcd_sect5(void);
151     void dcd_sect6(void);
152     void dcd_sect7(void);
153
154     void ini2date(char mmyyyy[]);
```

```
155  void str_elem(char celem[]);
156  void out_data(void);
157  void out_grads(void);
158
159  /*=========================================================================*/
160  /* MAIN PROGRAM                                                            */
161  /*=========================================================================*/
162  main(argc,argv)
163  int       argc;
164  char      **argv;
165  {
166       unsigned long      ns;
167       unsigned long      wkl;
168       char               wk[105],cns[5];
169            /*    Input File    */
170            if(2 != argc)
171                      {
172                      usage();
173                      exit(0);
174                      }
175            if(NULL == (fp = fopen(argv[1],"rb")))
176                      {
177                      printf("¥nCannot open FILE : %s¥n",argv[1]);
178                      usage();
179                      exit(1);
180                      }
181
182            fname=argv[1];
183            recno=0;
184
185            /*       area file          */
186       if ( OUTASCII == 1){
187             if(NULL != (fpa = fopen("grib2area.txt","rt")))
188                      {
189                      fgets(wk,100,fpa);
190                      sscanf(wk,"%ld%ld%ld%ld%",&ii1,&ii2,&jj1,&jj2);
191                          if ( 0 == isdigit(ii1))
192                                   sscanf(wk,"%s%ld%ld%ld%ld%",cns,&ii1,&ii2,&jj1,&jj2);
193                      }
194            else     {
195                      ii1=53;ii2=57;jj1=21;jj2=25;
196                      printf("Can't Open Area File !");
197                      printf("OutPut Japan Area (default)!!");
198                      }
199            printf(" ii1,ii2,jj1,jj2=%d %d %d %d¥n",ii1,ii2,jj1,jj2);
200            }
201  /*-------------------------------------------------------------------------*/
202  /*     DECODE SECT 0   : Indicator Section                                 */
203  /*-------------------------------------------------------------------------*/
204       dcd_sect0();
205  /*-------------------------------------------------------------------------*/
206  /*     DECODE SECT 1   : Identification Section                            */
207  /*-------------------------------------------------------------------------*/
```

```
208         printf("¥n<<SECTION 1>> : Identification Section ¥n");
209         dcd_sect1();
210     /*-------------------------------------------------------------------------*/
211     /*     DECODE SECT 2     : (Local Use Section)                             */
212     /*-------------------------------------------------------------------------*/
213         if( fread(buffer,sizeof(char), 5,fp) != 5 ){
214             printf("Read ERR SECT 2 !! File(%s)¥n",fname);
215             exit(72);
216             }
217         len2=convlong(&buffer[0], 4);
218         ns2=convlong(&buffer[4], 1);
219         fseek(fp,-5,1);                 /*   Back Space 5 bytes   */
220             switch(ns2){
221                 case 2:
222                     printf("<<SECTION 2>> : (Local Use Section)¥n");
223                     goto SECT2;
224                 case 3:
225                     printf("¥n<<SECTION 3>> : Grid Definition Section¥n");
226                     goto SECT3;
227                 default:
228                     printf("ERROR in Section 8 !!");
229                     printf(" len ns = %d %d¥n",wkl,buffer[0]);
230                     exit(99);
231             }
232     SECT2:
233         dcd_sect2();
234     /*-------------------------------------------------------------------------*/
235     /*     DECODE SECT 3       : Grid Definition Section                       */
236     /*-------------------------------------------------------------------------*/
237         printf("¥n<<SECTION 3>> : Grid Definition Section¥n");
238     SECT3:
239         dcd_sect3();
240     /*-------------------------------------------------------------------------*/
241     /*     DECODE SECT 4         : Product Definition Section                  */
242     /*-------------------------------------------------------------------------*/
243         printf("¥n<<SECTION 4>> : Product Definition Section¥n");
244     SECT4:
245         dcd_sect4();
246     /* Identify Data(elm lvl date period */
247         identifydata();
248     /*-------------------------------------------------------------------------*/
249     /*     DECODE SECT 5         : Data Representation Section                 */
250     /*-------------------------------------------------------------------------*/
251         printf("¥n<<SECTION 5>> : Data Representation Section¥n");
252         dcd_sect5();
253     /*-------------------------------------------------------------------------*/
254     /*     DECODE SECT 6           : Bit-map Section                           */
255     /*-------------------------------------------------------------------------*/
256         printf("¥n<<SECTION 6>> : Bit-map Section¥n");
257         dcd_sect6();
258     /*-------------------------------------------------------------------------*/
259     /*     DECODE SECT 7            : Data Section                             */
260     /*-------------------------------------------------------------------------*/
```

```
261         printf("¥n<<SECTION 7>> : Data Section¥n");
262         dcd_sect7();
263         if( IFOUT      == 1 ) out_data();                          /*    OutPut Data   */
264         if( OUTGRADS == 1 ) out_grads();                           /*    OutPut GRADS */
265     /*-------------------------------------------------------------------------*/
266     /*    DECODE SECT 8          : END or LOOP ?                                */
267     /*-------------------------------------------------------------------------*/
268         if( fread(buffer,sizeof(char), 4,fp) != 4 ){
269             printf("Read ERR !! File(%s)¥n",argv[1]);
270             exit(70);
271         }
272         if(buffer[0] == '7' && buffer[1] == '7' &&   buffer[2] == '7' && buffer[3] == '7'){
273             printf("GRIB END !!");
274             exit(0);
275             }
276         else{
277             wkl=convlong(&buffer[0], 4);
278             if( fread(buffer,sizeof(char), 1,fp) != 1 ){
279             printf("Read ERR !! File(%s)¥n",argv[1]);
280             exit(71);
281             }
282             fseek(fp,-5,1);              /*   Back Space 5 bytes   */
283             ns=convlong(&buffer[0], 1);
284             switch(ns){
285                 case 2:
286                     printf("¥n----------¥n");
287                     printf("Record No=%d ¥n",++recno);
288                     printf("----------¥n");
289                     printf("<<SECTION 2>> : (Local Use Section)¥n");
290                     goto SECT2;
291                 case 3:
292                     printf("¥n----------¥n");
293                     printf("Record No=%d ¥n",++recno);
294                     printf("----------¥n");
295                     printf("¥n<<SECTION 3>> : Grid Definition Section¥n");
296                     goto SECT3;
297                 case 4:
298                     printf("¥n----------¥n");
299                     printf("Record No=%d ¥n",++recno);
300                     printf("----------¥n");
301                     printf("¥n<<SECTION 4>> : Product Definition Section¥n");
302                     goto SECT4;
303                 default:
304                     printf("ERROR in Section 8 !!");
305                     printf(" len ns = %d %d¥n",wkl,buffer[0]);
306                     exit(99);
307             }
308         }
309     }
310     /*=========================================================================*/
311     /*       END MAIN()       */
312     /*=========================================================================*/
313     /*    DEFINE FUNCTIONS                                                     */
```

```
314  /*=========================================================================*/
315  /*-------------------------------------------------------------------------*/
316  /*     DECODE SECTION 0                                                    */
317  /*-------------------------------------------------------------------------*/
318  void dcd_sect0(void)
319  {
320  /*     DECODE SECT 0                                                       */
321       /*   Check Header   */
322            if( fread(buffer,sizeof(char), 4, fp) == 0 ) exit(2);
323            while( buffer[0] != 'G' || buffer[1] != 'R' ||
324                      buffer[2] != 'I' || buffer[3] != 'B' ){
325                       printf("Cannot Find GRIB header !!¥n");
326
327                       fclose(fp);exit(99);
328                       }
329            printf("----------------------¥n");
330            printf("---- GRIB FILE !! -----¥n");
331            printf("----------------------¥n");
332       fread(buffer,sizeof(char), 12, fp);
333       mtn=convlong(&buffer[2], 1);
334       printf("¥n<<SECTION 0>> : Indicator Section¥n");
335       printf("GRIB Master Table Number : %d¥n",mtn);
336       printf("GRIB Edition Number          : %d¥n",buffer[3]);
337       printf("buffer[4]= %d¥n",convlong(&buffer[4],4));
338       if( convlong(&buffer[4],4)   == 0){
339            len = convlong(&buffer[8], 4);
340            printf("Total length of GRIB        : %d¥n",len);
341            }
342       else
343            {
344            printf("Larg Record !! Not Supported !!¥n¥n");
345            exit(3);
346            }
347  }
348
349  /*-------------------------------------------------------------------------*/
350  /*     DECODE SECTION 1                                                    */
351  /*-------------------------------------------------------------------------*/
352  void dcd_sect1(void)
353  {
354       unsigned long      ns1,idcenter,icsubc;
355
356       fread(buffer,sizeof(char), 5,fp);
357       len1=convlong(&buffer[0], 4);
358       ns1=convlong(&buffer[4], 1);
359       fread(buffer,sizeof(char),len1-5,fp);
360       idcenter=convlong(&buffer[0], 2);
361       icsubc=convlong(&buffer[2],2);
362       ltvn=convlong(&buffer[5],1);
363       iyyyy=convlong(&buffer[7], 2);
364       imm=convlong(&buffer[9], 1);
365       iday=convlong(&buffer[10], 1);
366       printf("Length of Section 1   : %d¥n",len1);
```

```
367         printf("Number of Section      : %d¥n",ns1);
368         printf("Identification of centre       : %d¥n",idcenter);
369         printf("Identification of sub-centre : %d¥n",icsubc);
370         printf("GRIB Master Tables Version Number : %d¥n",buffer[4]);
371         printf("GRIB Local Tables Version Number   : %d¥n",buffer[5]);
372         printf("Significance of Reference Time      : %d¥n",buffer[6]);
373         printf("Year                          : %d¥n",iyyyy);
374         printf("Month                          : %d¥n",imm);
375         printf("Day                            : %d¥n",iday);
376         printf("Hour                           : %d¥n",buffer[11]);
377         printf("Minute                         : %d¥n",buffer[12]);
378         printf("Second                         : %d¥n",buffer[13]);
379         printf("Production status of processed data : %d¥n",buffer[14]);
380         printf("Type of processed data                      : %d¥n",buffer[15]);
381
382         printf("-----------------------¥n");
383         printf("Record No=%d ¥n",++recno);
384         printf("-----------------------¥n");
385     }
386
387     /*---------------------------------------------------------------------------*/
388     /*     DECODE SECTION 2                                                      */
389     /*---------------------------------------------------------------------------*/
390     void dcd_sect2(void)
391     {
392         if( fread(buffer,sizeof(char), 5,fp) != 5 ){
393             printf("Read ERR SECT 2 !! File(%s)¥n",fname);
394             exit(72);
395             }
396         len2=convlong(&buffer[0], 4);
397         ns2=convlong(&buffer[4], 1);
398         printf("Length of Section 2    : %d¥n",len2);
399         printf("Number of Section      : %d¥n",ns2);
400         if( len2-4 != 0 ){
401             fread(buffer,sizeof(char),len2-5,fp);
402             printf(" Local use .... Not Supported!!");
403             exit(2);
404         }
405     }
406
407     /*---------------------------------------------------------------------------*/
408     /*     DECODE SECTION 3                                                      */
409     /*---------------------------------------------------------------------------*/
410     void dcd_sect3(void)
411     {
412         unsigned long wkl,i,ol,Ntoe;
413
414         if( fread(buffer,sizeof(char), 5,fp) != 5 ){
415             printf("Read ERR SECT 3 !! File(%s)¥n",fname);
416             exit(73);
417             }
418         len3=convlong(&buffer[0], 4);
419         ns3=convlong(&buffer[4], 1);
```

```
420
421         fread(buffer,sizeof(char),len3-5,fp);
422         ijmx = convlong(&buffer[1], 4);
423         Ngdt = convlong(&buffer[7], 2);
424         ol     = convlong(&buffer[5], 1);
425         printf("Length of section                    : %d¥n",len3);
426         printf("Number of section                    : %d¥n",ns3);
427         printf("Source of grid definition            : %d¥n",buffer[0]);
428         printf("Number of data points                : %d¥n",ijmx);
429         printf("optional list                        : %d¥n",ol);
430         printf("Interpretation of list               : %d¥n",buffer[6]);
431         printf("Grid Definition Template Number : %d¥n",Ngdt);
432         switch(Ngdt){
433             case      0:
434                 printf("((Grid Definition Template 3.0))¥n");
435                 printf("Shape of the earth                                    : %d¥n",buffer[9]);
436                 printf("Scale factor of radius of spherical earth             : %d¥n",buffer[10]);
437                 wkl = convlong(&buffer[11], 4);
438                 printf("Scaled value of radius of spherical earth             : %d¥n",wkl);
439                 printf("Scale factor of major axis of oblate spheroid earth   : %d¥n",buffer[15]);
440                 wkl =   convlong(&buffer[16], 4);
441                 printf("Scaled value of major axis of oblate spheroid earth   : %d¥n",wkl);
442                 printf("Scale factor of minor axis of oblate spheroid earth   : %d¥n",buffer[20]);
443                 wkl = convlong(&buffer[21], 4);
444                 printf("Scaled value of minor axis of oblate spheroid earth   : %d¥n",wkl);
445                 Ni   = convlong(&buffer[25], 4);
446                 Nj   = convlong(&buffer[29], 4);
447                 printf("number of points along a parallel                     : %d¥n",Ni);
448                 printf("number of points along a meridian                     : %d¥n",Nj);
449                 wkl = convlong(&buffer[33], 4);
450                 printf("Basic angle of the initial production domain          : %d¥n",wkl);
451                 wkl = convlong(&buffer[37], 4);
452                 printf("Subdivisions of basic angle                           : %d¥n",wkl);
453                 La1 = convlong(&buffer[41], 4);
454                 Lo1 = convlong(&buffer[45], 4);
455                 printf("latitude of first grid point(La1)                     : %d¥n",La1);
456                 printf("longitude of first grid point(Lo1)                    : %d¥n",Lo1);
457                 printf("Resolution and component flags                        : %d¥n",buffer[49]);
458                 La2 = convlatlon(&buffer[50]);
459                 Lo2 = convlatlon(&buffer[54]);
460                 printf("latitude of last grid point(La2)                      : %d¥n",La2);
461                 printf("longitude of last grid point(Lo2)                     : %d¥n",Lo2);
462                 Di   = convlong(&buffer[58], 4);
463                 Dj   = convlong(&buffer[62], 4);
464                 printf("i direction increment(Di)                             : %d¥n",Di);
465                 printf("j direction increment(Dj)                             : %d¥n",Dj);
466                 printf("Scanning mode                                         : %d¥n",buffer[66]);
467                 break;
468             case 40:  /*   Not Checked   TEST TEST TEST   */
469                 printf("((Grid Definition Template 3.40))¥n");
470                 printf("Shape of the earth                                    : %d¥n",buffer[9]);
471                 printf("Scale factor of radius of spherical earth             : %d¥n",buffer[10]);
472                 wkl = convlong(&buffer[11], 4);
```

```
473                 printf("Scaled value of radius of spherical earth                : %d¥n",wkl);
474                 printf("Scale factor of major axis of oblate spheroid earth   : %d¥n",buffer[15]);
475                 wkl =   convlong(&buffer[16], 4);
476                 printf("Scaled value of major axis of oblate spheroid earth   : %d¥n",wkl);
477                 printf("Scale factor of minor axis of oblate spheroid earth   : %d¥n",buffer[20]);
478                 wkl = convlong(&buffer[21], 4);
479                 printf("Scaled value of minor axis of oblate spheroid earth   : %d¥n",wkl);
480                 Ni    = convlong(&buffer[25], 4);
481                 Nj    = convlong(&buffer[29], 4);
482                 printf("number of points along a parallel                        : %d¥n",Ni);
483                 printf("number of points along a meridian                         : %d¥n",Nj);
484                 wkl = convlong(&buffer[33], 4);
485                 printf("Basic angle of the initial production domain           : %d¥n",wkl);
486                 wkl = convlong(&buffer[37], 4);
487                 printf("Subdivisions of basic angle                              : %d¥n",wkl);
488                 La1 = convlong(&buffer[41], 4);
489                 Lo1 = convlong(&buffer[45], 4);
490                 printf("latitude of first grid point(La1)                      : %d¥n",La1);
491                 printf("longitude of first grid point(Lo1)                     : %d¥n",Lo1);
492                 printf("Resolution and component flags                           : %d¥n",buffer[49]);
493                 La2 = convlatlon(&buffer[50]);
494                 Lo2 = convlatlon(&buffer[54]);
495                 printf("latitude of last grid point(La2)                       : %d¥n",La2);
496                 printf("longitude of last grid point(Lo2)                      : %d¥n",Lo2);
497                 Di    = convlong(&buffer[58], 4);
498                 Ntoe= convlong(&buffer[62], 4);
499                 printf("i direction increment(Di)                                : %d¥n",Di);
500                 printf("Number of parallels from pole to equator(N)          : %d¥n",Ntoe);
501                 printf("Scanning mode                                              : %d¥n",buffer[66]);
502                 printf("Number of points along each grid line ¥n");
503                 printf("       Line      Points¥n");
504                 mxnp=0;
505                 for(i=0 ;i < Nj ;++i)
506                     {
507                     np[i]=convlong(&buffer[67+i*2], ol );
508                     if(np[i] > mxnp) mxnp=np[i];
509                     }
510                 break;
511             default      :
512                 printf("This Grid is Not Supported !!");
513                 exit(10);
514         }
515     }
516
517     /*---------------------------------------------------------------------*/
518     /*    DECODE SECTION 4                                                 */
519     /*---------------------------------------------------------------------*/
520     void dcd_sect4(void)
521     {
522         unsigned long      noc,tcut,svl1,svl2;
523         unsigned long      iyyyye,imme,idaye,Nt,nmiss,tinc;
524         unsigned long      i,latn,lats,lone,lonw,Nc,vstd,vdist;
525                     char       sfl1;
```

```
526
527         if( fread(buffer,sizeof(char), 5,fp) != 5 ){
528             printf("Read ERR SECT 4 !! File(%s)¥n",fname);
529             exit(74);
530             }
531         len4=convlong(&buffer[0], 4);
532         ns4=convlong(&buffer[4], 1);
533
534         fread(buffer,sizeof(char),len4-5,fp);
535         noc = convlong(&buffer[0], 2);
536         Npdt= convlong(&buffer[2], 2);
537         printf("Length of section 4                                    : %d¥n",len4);
538         printf("Number of section                                      : %d¥n",ns4);
539         printf("Number of coordinates values after Template    : %d¥n",noc);
540         printf("Product Definition Template Number                     : %d¥n",Npdt);
541         printf("(( Product Definition Template 4.%d ))¥n",Npdt);
542         switch(Npdt){
543             case       2:                          /*     PDT4.2      */
544                 pc       = convlong(&buffer[4], 1);
545                 pn       = convlong(&buffer[5], 1);
546                 tcut     = convlong(&buffer[9], 2);
547                 kt       = convlong(&buffer[13], 4);
548                 tys      = convlong(&buffer[17], 1);
549                 sfl1     = buffer[18];
550                 sfl1     = mem2char(&sfl1);
551                 svl1     = convlong(&buffer[19], 4);
552                 vl1      = svl1*pow(10.0,-1*(double)sfl1);
553                 svl2     = convlong(&buffer[25], 4);
554                 dfn      = convlong(&buffer[29], 1);
555                 nmem     = -1;
556                 printf("Parameter category                                     : %d¥n",pc);
557                 printf("Parameter number                                       : %d¥n",pn);
558                 printf("Type of generating process                             : %d¥n",buffer[6]);
559                 printf("Background generating process identifier               : %d¥n",buffer[7]);
560                 printf("Forecast generating process identifier                 : %d¥n",buffer[8]);
561                 printf("Hours after reference time of data cut-off             : %d¥n",tcut);
562                 printf("Minutes after reference time of data cut-off           : %d¥n",buffer[11]);
563                 printf("Indicator of unit of time range                        : %d¥n",buffer[12]);
564                 printf("Forecast time in units defined by octet 18             : %d¥n",kt);
565                 printf("Type of first fixed surface                            : %d¥n",tys);
566                 printf("Scale factor of first fixed surface                    : %d¥n",sfl1);
567                 printf("Scaled value of first fixed surface                    : %d¥n",svl1);
568                 printf("Type of second fixed surface                           : %d¥n",buffer[23]);
569                 printf("Scale factor of second fixed surface                   : %d¥n",buffer[24]);
570                 printf("Scaled value of second fixed surface                   : %d¥n",svl2);
571                 printf(" Derived forecast (see Code Table 4.7)                 : %d¥n",dfn);
572                 printf("Number of forecasts in ensemble                        : %d¥n",buffer[30]);
573                 break;
574             case      11:                           /*     PDT4.11     */
575                 pc       = convlong(&buffer[4], 1);
576                 pn       = convlong(&buffer[5], 1);
577                 tcut     = convlong(&buffer[9], 2);
578                 kt       = convlong(&buffer[13], 4);
```

```
579                 tys      = convlong(&buffer[17], 1);
580                 sfl1     = buffer[18];
581                 sfl1     = mem2char(&sfl1);
582                 svl1     = convlong(&buffer[19], 4);
583                 vl1      = svl1*pow(10.0,-1*(double)sfl1);
584                 svl2     = convlong(&buffer[25], 4);
585                 tyens    = convlong(&buffer[29], 1);
586                 nmem     = convlong(&buffer[30], 1);
587                 iyyyye = convlong(&buffer[32], 2);
588                 imme     = convlong(&buffer[34], 1);
589                 idaye    = convlong(&buffer[35], 1);
590                 Nt       = convlong(&buffer[39], 1);
591                 nmiss    = convlong(&buffer[40], 4);
592                 printf("Parameter category                                : %d¥n",pc);
593                 printf("Parameter number                                  : %d¥n",pn);
594                 printf("Type of generating process                        : %d¥n",buffer[6]);
595                 printf("Background generating process identifier          : %d¥n",buffer[7]);
596                 printf("Forecast generating process identifier            : %d¥n",buffer[8]);
597                 printf("Hours after reference time of data cut-off        : %d¥n",tcut);
598                 printf("Minutes after reference time of data cut-off      : %d¥n",buffer[11]);
599                 printf("Indicator of unit of time range                   : %d¥n",buffer[12]);
600                 printf("Forecast time in units defined by octet 18        : %d¥n",kt);
601                 printf("Type of first fixed surface                       : %d¥n",tys);
602                 printf("Scale factor of first fixed surface               : %d¥n",sfl1);
603                 printf("Scaled value of first fixed surface               : %d¥n",svl1);
604                 printf("Type of second fixed surface                      : %d¥n",buffer[23]);
605                 printf("Scale factor of second fixed surface              : %d¥n",buffer[24]);
606                 printf("Scaled value of second fixed surface              : %d¥n",svl2);
607                 printf("Type of ensemble forecast                         : %d¥n",buffer[29]);
608                 printf("Perturbation number                               : %d¥n",buffer[30]);
609                 printf("Number of forecasts in ensemble                   : %d¥n",buffer[31]);
610                 printf("Year                                    : %d¥n",iyyyye);
611                 printf("Month                                   : %d¥n",imme);
612                 printf("Day                                     : %d¥n",idaye);
613                 printf("Hour                                    : %d¥n",buffer[36]);
614                 printf("Minut                                   : %d¥n",buffer[37]);
615                 printf("Second                                  : %d¥n",buffer[38]);
616                 printf("n - Number of time range        : %d¥n",Nt);
617                 printf("Total number of data values missing               : %d¥n",nmiss);
618                 for(i=0 ; i < Nt ; ++i){
619                     lt       = convlong(&buffer[47+i*12], 4);
620                     tinc     = convlong(&buffer[52+i*12], 4);
621                     printf("Statistical process used to calculate ....    : %d¥n",buffer[44+i*12]);
622                     printf("Type of time increment                            : %d¥n",buffer[45+i*12]);
623                     printf("Indicator of unit of time for time range      : %d¥n",buffer[46+i*12]);
624                     printf("Length of the time range                          : %d¥n",lt);
625                     printf("Indicator of unit of time for the increment   : %d¥n",buffer[51+i*12]);
626                     printf("Time increment between successive fields      : %d¥n",tinc);
627                 }
628                 break;
629             case    12:                     /*    PDT4.12    */
630                 pc       = convlong(&buffer[4], 1);
631                 pn       = convlong(&buffer[5], 1);
```

```
632                 tcut     = convlong(&buffer[9], 2);
633                 kt       = convlong(&buffer[13], 4);
634                 tys      = convlong(&buffer[17], 1);
635                 sfl1     = buffer[18];
636                 sfl1     = mem2char(&sfl1);
637                 svl1     = convlong(&buffer[19], 4);
638                 vl1      = svl1*pow(10.0,-1*(double)sfl1);
639                 svl2     = convlong(&buffer[25], 4);
640                 dfn      = convlong(&buffer[29], 1);
641                 nmem     = -1;
642                 iyyyye = convlong(&buffer[31], 2);
643                 imme     = convlong(&buffer[33], 1);
644                 idaye    = convlong(&buffer[34], 1);
645                 Nt       = convlong(&buffer[38], 1);
646                 nmiss    = convlong(&buffer[39], 4);
647                 printf("Parameter category                                : %d¥n",pc);
648                 printf("Parameter number                                  : %d¥n",pn);
649                 printf("Type of generating process                        : %d¥n",buffer[6]);
650                 printf("Background generating process identifier          : %d¥n",buffer[7]);
651                 printf("Forecast generating process identifier            : %d¥n",buffer[8]);
652                 printf("Hours after reference time of data cut-off        : %d¥n",tcut);
653                 printf("Minutes after reference time of data cut-off      : %d¥n",buffer[11]);
654                 printf("Indicator of unit of time range                   : %d¥n",buffer[12]);
655                 printf("Forecast time in units defined by octet 18        : %d¥n",kt);
656                 printf("Type of first fixed surface                       : %d¥n",tys);
657                 printf("Scale factor of first fixed surface               : %d¥n",sfl1);
658                 printf("Scaled value of first fixed surface               : %d¥n",svl1);
659                 printf("Type of second fixed surface                      : %d¥n",buffer[23]);
660                 printf("Scale factor of second fixed surface              : %d¥n",buffer[24]);
661                 printf("Scaled value of second fixed surface              : %d¥n",svl2);
662                 printf(" Derived forecast (see Code Table 4.7)            : %d¥n",dfn);
663                 printf("Number of forecasts in ensemble                   : %d¥n",buffer[30]);
664                 printf("Year                                              : %d¥n",iyyyye);
665                 printf("Month                                             : %d¥n",imme);
666                 printf("Day                                               : %d¥n",idaye);
667                 printf("Hour                                              : %d¥n",buffer[35]);
668                 printf("Minut                                             : %d¥n",buffer[36]);
669                 printf("Second                                            : %d¥n",buffer[37]);
670                 printf("n - Number of time range                          : %d¥n",Nt);
671                 printf("Total number of data values missing               : %d¥n",nmiss);
672                 for(i=0 ; i < Nt ; ++i){
673                     lt       = convlong(&buffer[46+i*12], 4);
674                     tinc     = convlong(&buffer[51+i*12], 4);
675                     printf("Statistical process used to calculate ....    : %d¥n",buffer[43+i*12]);
676                     printf("Type of time increment                        : %d¥n",buffer[44+i*12]);
677                     printf("Indicator of unit of time for time range      : %d¥n",buffer[45+i*12]);
678                     printf("Length of the time range                      : %d¥n",lt);
679                     printf("Indicator of unit of time for the increment   : %d¥n",buffer[50+i*12]);
680                     printf("Time increment between successive fields      : %d¥n",tinc);
681                 }
682                 break;
683             case    13:                 /*    PDT4.13    */
684                 pc       = convlong(&buffer[4], 1);
```

```
685                 pn       = convlong(&buffer[5], 1);
686                 tcut     = convlong(&buffer[9], 2);
687                 kt       = convlong(&buffer[13], 4);
688                 tys      = convlong(&buffer[17], 1);
689                 sfl1     = buffer[18];
690                 sfl1     = mem2char(&sfl1);
691                 svl1     = convlong(&buffer[19], 4);
692                 vl1      = svl1*pow(10.0,-1*(double)sfl1);
693                 svl2     = convlong(&buffer[25], 4);
694                 dfn      = convlong(&buffer[29], 1);
695                 nmem     = -1;
696                 latn     = convlong(&buffer[36], 4);
697                 lats     = convlong(&buffer[40], 4);
698                 lone     = convlong(&buffer[44], 4);
699                 lonw     = convlong(&buffer[48], 4);
700                 Nc       = convlong(&buffer[52], 1);
701                 vstd     = convlong(&buffer[54], 4);
702                 vdist    = convlong(&buffer[59], 4);
703                 iyyyye = convlong(&buffer[63], 2);
704                 imme     = convlong(&buffer[65], 1);
705                 idaye    = convlong(&buffer[66], 1);
706                 Nt       = convlong(&buffer[70], 1);
707                 nmiss    = convlong(&buffer[71], 4);
708                 printf("Parameter category                                  : %d¥n",pc);
709                 printf("Parameter number                                    : %d¥n",pn);
710                 printf("Type of generating process                          : %d¥n",buffer[6]);
711                 printf("Background generating process identifier            : %d¥n",buffer[7]);
712                 printf("Forecast generating process identifier              : %d¥n",buffer[8]);
713                 printf("Hours after reference time of data cut-off          : %d¥n",tcut);
714                 printf("Minutes after reference time of data cut-off        : %d¥n",buffer[11]);
715                 printf("Indicator of unit of time range                     : %d¥n",buffer[12]);
716                 printf("Forecast time in units defined by octet 18          : %d¥n",kt);
717                 printf("Type of first fixed surface                         : %d¥n",tys);
718                 printf("Scale factor of first fixed surface                 : %d¥n",sfl1);
719                 printf("Scaled value of first fixed surface                 : %d¥n",svl1);
720                 printf("Type of second fixed surface                        : %d¥n",buffer[23]);
721                 printf("Scale factor of second fixed surface                : %d¥n",buffer[24]);
722                 printf("Scaled value of second fixed surface                : %d¥n",svl2);
723                 printf("Derived forecast (see Code Table 4.7)               : %d¥n",dfn);
724                 printf("Number of forecasts in the ensemble (N)             : %d¥n",buffer[30]);
725                 printf("Cluster identifier                                  : %d¥n",buffer[31]);
726                 printf("Number of cluster to which the high resolution control
727     belongs : %d¥n",buffer[32]);
728                 printf("Number of cluster to which the low resolution control
729     belongs   : %d¥n",buffer[33]);
730                 printf("Total number of clusters                            : %d¥n",buffer[34]);
731                 printf("Clustering method (see Code Table 4.8)              : %d¥n",buffer[35]);
732                 printf("Northern latitude of cluster domain                 : %d¥n",latn);
733                 printf("Southern latitude of cluster domain                 : %d¥n",lats);
734                 printf("Eastern longitude of cluster domain                 : %d¥n",lone);
735                 printf("Western longitude of cluster domain                 : %d¥n",lonw);
736                 printf("Number of forecasts in the cluster                  : %d¥n",Nc);
737                 printf("Scale factor of standard deviation in the cluster   : %d¥n",buffer[53]);
```

```
738                 printf("Scaled value of standard deviation in the cluster               : %d¥n",vstd);
739                 printf("Scale factor of distance of the cluster from ensemble mean   : %d¥n",buffer[58]);
740                 printf("Scaled value of distance of the cluster from ensemble mean    : %d¥n",vdist);
741                 printf("Year                                    : %d¥n",iyyyye);
742                 printf("Month                                    : %d¥n",imme);
743                 printf("Day                                      : %d¥n",idaye);
744                 printf("Hour                                     : %d¥n",buffer[67]);
745                 printf("Minut                                    : %d¥n",buffer[68]);
746                 printf("Second                                   : %d¥n",buffer[69]);
747                 printf("n - Number of time range        : %d¥n",Nt);
748                 printf("Total number of data values missing                 : %d¥n",nmiss);
749                 for(i=0 ; i < Nt ; ++i){
750                     lt       = convlong(&buffer[78+i*12], 4);
751                     tinc     = convlong(&buffer[83+i*12], 4);
752                     printf("Statistical process used to calculate ....          : %d¥n",buffer[75+i*12]);
753                     printf("Type of time increment                                  : %d¥n",buffer[76+i*12]);
754                     printf("Indicator of unit of time for time range            : %d¥n",buffer[77+i*12]);
755                     printf("Length of the time range                                : %d¥n",lt);
756                     printf("Indicator of unit of time for the increment        : %d¥n",buffer[82+i*12]);
757                     printf("Time increment between successive fields             : %d¥n",tinc);
758                 }
759                     printf("List of Nc ensemble forecast numbers :");
760                 for(i=0 ; i < Nc ; ++i){
761                     printf(" %2d",buffer[74+Nt*12+i+1]);
762                 }
763                     printf("¥n");
764                 break;
765             default:
766                 printf("This Product Definition Template is Not Supported !!");
767                 exit(11);
768         }
769     }
770     /*--------------------------------------------------------------------------*/
771     /*     DECODE SECTION 5                                                     */
772     /*--------------------------------------------------------------------------*/
773     void dcd_sect5(void)
774     {
775         fread(buffer,sizeof(char), 5,fp);
776         len5=convlong(&buffer[0], 4);
777         ns5=convlong(&buffer[4], 1);
778         fread(buffer,sizeof(char),len5-5,fp);
779         ijdim = convlong(&buffer[0], 4);
780         ntemplate = convlong(&buffer[4], 2);
781
782         printf("Length of section in octets                 : %d¥n",len5);
783         printf("Number of section                               : %d¥n",ns5);
784         printf("Number of data points                           : %d¥n",ijdim);
785         printf("Data Representation Template Number          : %d¥n",ntemplate);
786         switch(ntemplate){
787             case      0:
788                 R = convfloat(&buffer[6], 4);
789                 E = convlong(&buffer[10], 2);
790                     if(E > 32768) E = (E - 32768 )*(-1);
```

```
791                 D = convlong(&buffer[12], 2);
792                     if(D > 32768) D = (D - 32768 )*(-1);
793                 nbit = convlong(&buffer[14], 1);
794                 printf("Reference value (R)                    : %f¥n",R);
795                 printf("Binary scale factor (E)                : %d¥n",E);
796                 printf("Decimal scale factor (D)               : %d¥n",D);
797                 printf("Number of bits .....                   : %d¥n",nbit);
798                 printf("Type of original field values          : %d¥n",buffer[15]);
799                 break;
800             default:
801                     printf("This Data Representation Template is Not Supported!!");
802                     exit(12);
803             }
804     }
805
806     /*---------------------------------------------------------------------*/
807     /*    DECODE SECTION 6                                                 */
808     /*---------------------------------------------------------------------*/
809     void dcd_sect6(void)
810     {
811         int     i;
812
813         fread(buffer,sizeof(char), 5,fp);
814         len6=convlong(&buffer[0], 4);
815         ns6=convlong(&buffer[4], 1);
816         fread(buffer,sizeof(char), 1,fp);
817         ifbitmap = convlong(&buffer[0], 1);
818         printf("Length of section in octets            : %d¥n",len6);
819         printf("Number of section                      : %d¥n",ns6);
820         printf("Bit-map indicator                      : %d¥n",ifbitmap);
821         for( i = 0 ; i < SIZEOFBITBUF ;++i){
822             map[i]=0xFF;
823             }
824         switch(ifbitmap){
825         case        0:
826             if ( len6-6 != fread(map,sizeof(char),len6-6,fp)){
827                 printf("Read ERR SECT 6-1 !! File(%s)¥n",fname);
828                 exit(100);
829                 }
830             break;
831             case    255:
832                 printf("No BITMAP !!¥n");
833                 break;
834             default:
835                 printf("This Bit-map indicator is Not Supportted!!");
836                 exit(13);
837         }
838     }
839
840     /*---------------------------------------------------------------------*/
841     /*    DECODE SECTION 7                                                 */
842     /*---------------------------------------------------------------------*/
843     void dcd_sect7(void)
```

```
844     {
845         unsigned char       msk_bit=MASK1;
846         unsigned long       posi,lgpv,i;
847
848
849         fread(buffer,sizeof(char), 5,fp);
850         len7=convlong(&buffer[0], 4);
851         ns7=convlong(&buffer[4], 1);
852         printf("Length of section in octets                    : %d¥n",len7);
853         printf("Number of section                                 : %d¥n",ns7);
854         fread(buffer2,sizeof(char),len7-5,fp);
855         printf("ntemplate=%d¥n",ntemplate);
856         switch(ntemplate){
857             case 0:
858
859                 posi = 0;
860                 for( i = 0 ;i < ijmx ;i++ ){
861                     nb = i / 8;
862                     mb = i % 8;
863                     if( 0 ==    (map[nb] & ( msk_bit >> mb ))){
864                         data[i]=UNDEF;
865                         }
866                     else{
867                         lgpv=longbybit(buffer2,&posi,nbit);
868                         data[i] = (R + lgpv*pow(2.0,(double)E))/pow(10.0,(double)D);
869                         }
870                     }
871
872                 printf("Check DATA !!       ¥n");
873                 printf("     No        value ¥n");
874                 for( i = 0; i < 10 ; ++i){
875                     printf("    %5d       %f¥n",i+1,data[i]);
876                 }
877                 printf("¥n¥n");
878                 for( i = ijmx-10; i < ijmx ; ++i){
879                     printf("    %5d       %f¥n",i+1,data[i]);
880                 }
881
882                 break;
883             default:
884                 printf("Unsupported Template !!");
885                 exit(14);
886         }
887     }
888     /*-------------------------------------------------------------------------*/
889     /*     OUT_DATA                                                            */
890     /*-------------------------------------------------------------------------*/
891     void out_data(void)
892     {
893         int                 itot;
894         unsigned int        i,j,ii,flen;
895         FILE                  *fpc1;
896         char                flc1[31];
```

```
897         char                celem[100],mmyyyy[7];
898         float               fla1,fdi,flo1,fdj;
899
900         if(( fpg = fopen(fl,"wb")) == NULL ){
901             printf("Stop, Create File(%s)¥n",fl);
902             exit(99);
903             }
904
905         switch(Ngdt)
906             {
907             case 0:
908                 if( fwrite(data,4,ijmx,fpg) == ijmx )
909                 {
910                     /*    Out Put GrADS Control File   */
911                     strcpy(flc1,fl);
912                     flen=strlen(flc1);
913                     flc1[flen      ]='.';flc1[flen + 1]='c';flc1[flen + 2]='t';
914                     flc1[flen + 3]='l';flc1[flen + 4]=0x00;
915                     if(( fpc1 = fopen(flc1,"wt")) == NULL ){
916                         printf("Stop, Create File(%s)¥n",flc);
917                         exit(98);
918                         }
919                     fla1=(-1)*(float)La1/1000000.0 ;fdi=(float)Di/1000000.0;
920                     flo1=(float)Lo1/1000000.0 ;fdj=(float)Dj/1000000.0;
921
922                     ini2date(mmyyyy);
923                     str_elem(celem);
924                     tcount=1;
925
926                     fprintf(fpc1,"dset ^%s¥n",fl);
927                     fprintf(fpc1,"undef -19999.¥n");
928                     fprintf(fpc1,"xdef    %d    linear    %f   %f¥n",Ni,flo1,fdi);
929                     fprintf(fpc1,"ydef    %d     linear   %f   %f¥n",Nj,fla1,fdj);
930                     fprintf(fpc1,"zdef    1      linear   %s   1 ¥n",clvl);
931                     fprintf(fpc1,"tdef    %d       linear    01%s   1mo¥n",tcount,mmyyyy);
932                     fprintf(fpc1,"vars    1 ¥n");
933                     fprintf(fpc1,"%s¥n",celem);
934                     fprintf(fpc1,"endvars¥n");
935                     fprintf(fpc1,"options yrev¥n");
936                     printf("Close Control File= %s ¥n",flc1);
937                     if ( fclose(fpc1) != 0 ){
938                         printf("Close ERR !! File(%s)¥n",flc1);
939                         exit(96);
940                         }
941                     }
942                 else{
943                     printf("Write ERR !! File(%s)¥n",fl);
944                     exit(98);
945                     }
946                 break;
947             default:
948                 printf(" This GDT is not supported !!   Ngdt= %d ¥n",Ngdt);
949                 exit(95);
```

```
950             }
951         if ( fclose(fpg) != 0 ){
952             printf("Close ERR !! File(%s)¥n",fl);
953             exit(97);
954             }
955     }
956
957     void out_grads(void)
958     {
959         int                 itot,flen;
960         unsigned int        i,j,ii,fmon;
961         char                cwk[5];
962         char                celem[100],mmyyyy[7];
963         float               fla1,fdi,flo1,fdj;
964
965         /*      3month or 1month     average */
966         if ( lt <= 31 && lt >= 28 ){
967             fmon=1;
968             }
969         else if ( lt <=92 && lt >= 89){
970             fmon=3;
971            }
972         else{
973            printf("Forecast range Not Supported !! lt= %d ",lt);
974            exit(90);
975             }
976
977         /*     Open file       */
978         if ( tcount == 0 )
979             {
980             strcpy(flg,fname);
981             flen=strlen(flg);
982             sprintf(cwk,"%1d",fmon);
983             flg[flen - 7]='_';flg[flen - 6]=cwk[0];flg[flen - 5]='m';
984             flg[flen - 4]='.';flg[flen - 3]='d'     ;flg[flen - 2]='a';
985             flg[flen - 1]='t';flg[flen - 0]=0x00;
986
987             if(( fpg = fopen(flg,"wb")) == NULL ){
988                 printf("Stop, Create File(%s)¥n",flg);
989                 exit(99);
990                 }
991             strcpy(flc,flg);
992             flc[flen - 3]='c';flc[flen - 2]='t'     ;flc[flen - 1]='l';
993             if(( fpc = fopen(flc,"wt")) == NULL ){
994                 printf("Stop, Create File(%s)¥n",flc);
995                 exit(99);
996                 }
997             }
998
999         /*      Write File     */
1000        if(Ngdt == 0 )
1001            {
1002                if( fwrite(data,4,ijmx,fpg) != ijmx )
```

```
1003                    {
1004                    printf("Write ERR !! File(%s)¥n",fl);
1005                    exit(98);
1006                    }
1007                tcount=tcount+1;
1008                printf("Wrote File= %s tcount=%d ¥n",flg,tcount);
1009            }
1010        else
1011            {
1012                printf(" This GDT is not sopported !!   Ngdt= %d ¥n",Ngdt);
1013                exit(95);
1014            }
1015
1016        /*    Close File    */
1017        if ( tcount == 1 && fmon == 3 || tcount == 3 && fmon == 1){
1018            if ( fclose(fpg) != 0 ){
1019                printf("Close ERR !! File(%s)¥n",flg);
1020                exit(97);
1021                }
1022
1023        /*    GrADS control File    */
1024            fla1=(-1)*(float)La1/1000000.0 ;fdi=(float)Di/1000000.0;
1025            flo1=(float)Lo1/1000000.0 ;fdj=(float)Dj/1000000.0;
1026
1027            ini2date(mmyyyy);
1028            str_elem(celem);
1029
1030            fprintf(fpc,"dset ^%s¥n",flg);
1031            fprintf(fpc,"undef -19999.¥n");
1032            fprintf(fpc,"xdef    %d    linear    %f   %f¥n",Ni,flo1,fdi);
1033            fprintf(fpc,"ydef    %d     linear   %f   %f¥n",Nj,fla1,fdj);
1034            fprintf(fpc,"zdef    1      linear   %s     1 ¥n",clvl);
1035            fprintf(fpc,"tdef    %d      linear    01%s   1mo¥n",tcount,mmyyyy);
1036            fprintf(fpc,"vars    1 ¥n");
1037            fprintf(fpc,"%s¥n",celem);
1038            fprintf(fpc,"endvars¥n");
1039            fprintf(fpc,"options yrev¥n");
1040
1041            if ( fclose(fpc) != 0 ){
1042                printf("Close ERR !! File(%s)¥n",flc);
1043                exit(96);
1044                }
1045             tcount=0;
1046             printf("Close Control File= %s ¥n",flc);
1047             }
1048    }
1049    /*=========================================================================*/
1050    void ini2date(char mmyyyy[])
1051    {
1052        unsigned long yyyy1,mm1;
1053        char      yyyy[5],mm[3];
1054
1055        if       ( imm == 12){ strcpy(mm,"Jan");yyyy1=iyyyy+1;}
```

```
1056        else if( imm ==   1){ strcpy(mm,"Feb");yyyy1=iyyyy   ;}
1057        else if( imm ==   2){ strcpy(mm,"Mar");yyyy1=iyyyy   ;}
1058        else if( imm ==   3){ strcpy(mm,"Apr");yyyy1=iyyyy   ;}
1059        else if( imm ==   4){ strcpy(mm,"May");yyyy1=iyyyy   ;}
1060        else if( imm ==   5){ strcpy(mm,"Jun");yyyy1=iyyyy   ;}
1061        else if( imm ==   6){ strcpy(mm,"Jul");yyyy1=iyyyy   ;}
1062        else if( imm ==   7){ strcpy(mm,"Aug");yyyy1=iyyyy   ;}
1063        else if( imm ==   8){ strcpy(mm,"Sep");yyyy1=iyyyy   ;}
1064        else if( imm ==   9){ strcpy(mm,"Oct");yyyy1=iyyyy   ;}
1065        else if( imm == 10){ strcpy(mm,"Nov");yyyy1=iyyyy   ;}
1066        else if( imm == 11){ strcpy(mm,"Dec");yyyy1=iyyyy   ;}
1067        else{ printf(" Initial month is strange!! imm=%d¥n",imm); exit(55);}
1068
1069        sprintf(mmyyyy,"%s%04d",mm,yyyy1);
1070    }
1071
1072    void str_elem(char celem[])
1073    {
1074         char              elm[5],lvl[5],spr[5],comm[70],wstr[10];
1075         char              sprcomm[20];
1076         unsigned long vl1hpa;
1077         unsigned int   i,len1,len2,len3,len4,len5,len6;
1078
1079         sprcomm[0]=0x00;
1080         elm[0]=0x00;lvl[0]=0x00;spr[0]=0x00;
1081         vl1hpa=vl1*0.01;
1082
1083     /*--------------- LEVEL     ------------------------------------------*/
1084         if       ( tys == 1     ){strcpy(lvl,"surf")            ;strcpy(clvl,"1000");}
1085         else if( tys == 101 ){strcpy(lvl,"sea")             ;strcpy(clvl,"1000");}
1086         else if( tys == 103 ){strcpy(lvl,"2m")              ;strcpy(clvl,"1000");}
1087         else if( tys == 100 ){sprintf(lvl,"%01d",vl1hpa);sprintf(clvl,"%3d",vl1hpa);}
1088         else     {printf(" This level is Not Supported !! tys %d¥n",tys);
1089                   exit(43);}
1090     /*--------------- SPREAD ? ---------------------------------------*/
1091         if ( dfn == 4 ){
1092            strcpy(spr,"s");strcpy(sprcomm,"spread ");
1093           }
1094         else if( dfn == 5 ){
1095            strcpy(spr,"l");strcpy(sprcomm,"larg anomaly index ");
1096           }
1097
1098     /*--------------- ELEMENT ----------------------------------------*/
1099     if( mtn == 0)
1100           {
1101           if       ( pc == 0 && pn == 0 ){
1102             if( tys != 100 ){
1103               strcpy(elm,"t")    ;sprintf(comm,"Temperature %s at %s   [K]",sprcomm,lvl);}
1104             else{
1105               strcpy(elm,"t")    ;sprintf(comm,"Temperature at %s hPa [K]",lvl);}
1106             }
1107           else if( pc == 0 && pn == 9 ){
1108             if( tys != 100 ){
```

```
1109              strcpy(elm,"t")   ;strcpy(spr,"a");
1110              sprintf(comm,"Temprature anoaly at %s hPa [K]",lvl);}
1111            else{
1112              strcpy(elm,"t")   ;strcpy(spr,"a");
1113              sprintf(comm,"Temprature anoaly at %s [K]",lvl);}
1114            }
1115          else if( pc == 1 && pn == 0 ){
1116            strcpy(elm,"q")    ;
1117            sprintf(comm,"Specific humidity at %s hPa [kg/kg]",lvl);}
1118          else if( pc == 1 && pn == 1 ){
1119            strcpy(elm,"rh")   ;
1120            sprintf(comm,"Relative humidity %s at %s hPa [%%]",sprcomm,lvl);}
1121          else if( pc == 1 && pn == 8 ){
1122            strcpy(elm,"rr")   ;
1123            sprintf(comm,"Total precipitaion %s [mm]",sprcomm);}
1124          else if( pc == 1 && pn ==210){
1125            strcpy(elm,"rrd")   ;lvl[0]=0x00;
1126            sprintf(comm,"Daily mean precipitaion %s [mm]",sprcomm);}
1127          else if( pc == 1 && pn ==211){
1128            strcpy(elm,"rrd") ;strcpy(spr,"a");lvl[0]=0x00;
1129            sprintf(comm,"Daily mean precipitaion anomaly [mm]");}
1130          else if( pc == 1 && pn ==212){
1131            strcpy(elm,"q")   ;strcpy(spr,"a");
1132            sprintf(comm,"Specific humidity anomaly at %s hPa [kg/kg]",lvl);}
1133          else if( pc == 1 && pn ==213){
1134            strcpy(elm,"rh") ;strcpy(spr,"a");
1135            sprintf(comm,"Relative humidity anomaly at %s hPa [%%]",lvl);}
1136          else if( pc == 2 && pn == 2 ){
1137            strcpy(elm,"u")    ;
1138            sprintf(comm,"u wind %s at %s hPa [m/s]",sprcomm,lvl);}
1139          else if( pc == 2 && pn == 3 ){
1140            strcpy(elm,"v")    ;
1141            sprintf(comm,"v wind %s at %s hPa [m/s]",sprcomm,lvl);}
1142          else if( pc == 2 && pn == 4 ){
1143            strcpy(elm,"psi") ;
1144            sprintf(comm,"Stream function at %s hPa [m^2/s]",lvl);}
1145          else if( pc == 2 && pn == 5 ){
1146            strcpy(elm,"chi") ;
1147            sprintf(comm,"Velocity potential at %s hPa [m^2/s]",lvl);}
1148          else if( pc == 2 && pn ==210){
1149            strcpy(elm,"u")   ;strcpy(spr,"a");
1150            sprintf(comm,"u wind anomaly at %s hPa [m/s]",lvl);}
1151          else if( pc == 2 && pn ==211){
1152            strcpy(elm,"v")   ;strcpy(spr,"a");
1153            sprintf(comm,"v wind anomaly at %s hPa [m/s]",lvl);}
1154          else if( pc == 3 && pn == 0 ){
1155            if(tys != 100 ){
1156              strcpy(elm,"p")    ;
1157              sprintf(comm,"Pressure %s at %s   [Pa]",sprcomm,lvl);}
1158            else{
1159              strcpy(elm,"p")    ;
1160              sprintf(comm,"Pressure %s at %s hPa [Pa]",sprcomm,lvl);}
1161            }
```

```
1162            else if( pc == 3 && pn == 1 ){
1163              strcpy(elm,"p");
1164              sprintf(comm,"Sea level pressure %s [Pa]",sprcomm);}
1165            else if( pc == 3 && pn == 5 ){
1166              strcpy(elm,"z")     ;
1167              sprintf(comm,"Geopotential height %s at %s hPa [gpm]",sprcomm,lvl);}
1168            else if( pc == 3 && pn == 8 ){
1169              if( tys != 100 ){
1170                strcpy(elm,"p")   ; strcpy(spr,"a");
1171                sprintf(comm,"Pressure anomaly at %s [Pa]",lvl);}
1172              else{
1173                strcpy(elm,"p")   ; strcpy(spr,"a");
1174                sprintf(comm,"Pressure anomaly at %s hPa [Pa]",lvl);}
1175              }
1176            else if( pc == 3 && pn == 9 ){
1177              strcpy(elm,"z")   ;strcpy(spr,"a");
1178              sprintf(comm,"Geopotential height anomaly at %s hPa [gpm]",lvl);}
1179            else    {
1180              printf(" This element Not Supported !! mtn,pc,pn %d,%d,%d¥n",mtn,pc,pn); exit(40);}
1181            }
1182          else if(mtn == 10 )
1183            {
1184            if       ( pc == 3 && pn == 0 ){
1185              strcpy(elm,"sst") ;lvl[0]=0x00;
1186              sprintf(comm,"Sea surface temperature [K]");}
1187            else if( pc == 3 && pn ==192){
1188              strcpy(elm,"sst");strcpy(spr,"a");lvl[0]=0x00;
1189              sprintf(comm,"Sea surface temperature anomaly [K]");}
1190            else    {
1191              printf(" This element Not Supported !! mtn,pc,pn %d,%d,%d¥n",mtn,pc,pn); exit(41);}
1192            }
1193          else
1194            {
1195             printf(" This element Not Supported !! mtn,pc,pn %d,%d,%d¥n",mtn,pc,pn); exit(42);
1196             }
1197       /*---------- length of string & copy string ----------*/
1198          strcpy(wstr,"1 99 ** ");
1199
1200          len1=strlen(elm);
1201          len2=len1 + strlen(lvl);
1202          len3=len2 + strlen(spr);
1203          len4=8 ;
1204          len5=len4 + strlen(wstr);
1205          len6=len5 + strlen(comm);
1206
1207          for (i = 0     ; i < len1 ;++i)celem[i]=elm[i];
1208          for (i = len1; i < len2 ;++i)celem[i]=lvl[i-len1];
1209          for (i = len2; i < len3 ;++i)celem[i]=spr[i-len2];
1210          for (i = len3; i < len4 ;++i)celem[i]=' ';
1211          for (i = len4; i < len5 ;++i)celem[i]=wstr[i-len4];
1212          for (i = len5; i < len6 ;++i)celem[i]=comm[i-len5];
1213          celem[len6]=0x00;
1214
```

```
1215    /** printf("len1,len2,len3,len4,len5,len6=%d,%d,%d,%d,%d,%d¥n",len1,len2,len3,len4,len5,len6);
1216          printf(" elm = %s¥n",elm);
1217          printf(" lvl = %s¥n",lvl);
1218          printf(" apr = %s¥n",spr);
1219          printf(" wstr = %s¥n",wstr);
1220          printf(" comm = %s¥n",comm);
1221          printf(" ELEM= %s¥n",celem);
1222    **/
1223    }
1224
1225
1226    /*=====================================================================*/
1227    /*      TOOLS                                                          */
1228    /*=====================================================================*/
1229
1230    void usage(void)
1231    {
1232              puts("¥n");
1233              puts("Usage : GRIB2 <FILE-NAME>");
1234    }
1235
1236    long longbybit(unsigned char *buf,unsigned long *pos,unsigned long nbit)
1237    {
1238         char      i,ii;
1239         long      pos8,mg8,rem8,nbytes,remnbit;
1240         long                lout=0;
1241         long                lbig;
1242         unsigned char       *buf2,*cbig;
1243         unsigned char       mask,maskremn,full=0xff;
1244
1245         lbig=1;
1246         cbig = (unsigned char *)&lbig;
1247
1248         buf2 = (unsigned char *)&lout;
1249         mg8    = *pos % 8;
1250         remnbit= nbit % 8;
1251
1252         if( (*pos + nbit) % 8 == 0 ) rem8 = 0;
1253         else                                rem8 = 8 - (( *pos + nbit ) % 8 );
1254
1255         pos8 = ( *pos + nbit + rem8 ) / 8 - 1;
1256
1257         nbytes = ( nbit ) / 8;
1258         mask     = full >>    mg8 + rem8;
1259         for( i = 0;i < nbytes   ;i++ ){
1260              if( 1 == cbig[3] & 0x01 ) ii = 3 -i;
1261              else                           ii = i     ;
1262              buf2[ii] = ( buf[pos8 - i] >> rem8
1263                           | buf[pos8    - i - 1] << 8 - rem8 );
1264         }
1265
1266         if( 1 == cbig[3] & 0x01 ) ii = 3 - nbytes;
1267         else                           ii = nbytes ;
```

```
1268        if( remnbit <= 8 - rem8 ){
1269          maskremn = full >> 8 - remnbit;
1270          buf2[ii] = ( buf[pos8 - nbytes] >> rem8 & maskremn );
1271          }
1272        else{
1273          maskremn = full >> 8 - remnbit & full << 8 - rem8 ;
1274          buf2[ii] = ( buf[pos8 - nbytes] >> rem8
1275                       | buf[pos8 - nbytes -1] << 8 - rem8 & maskremn );
1276          }
1277        *pos = *pos + nbit;
1278        return(lout);
1279    }
1280
1281    long convlong( unsigned char *string, int nbyte )
1282    {
1283        int                 i,ii;
1284        long                numb=0;
1285        long                lbig;
1286        unsigned char       *chrwrk,*cbig;
1287
1288        lbig=1;
1289        cbig = (unsigned char *)&lbig;
1290        /*if( 1 == cbig[3] & 0x01 ) printf("BIG ENDIAN !!");*/
1291        chrwrk = (unsigned char *)&numb;
1292        for( i = 0; i < nbyte; i++ ) {
1293            if( 1 == cbig[3] & 0x01 ) ii = 4 - nbyte + i;
1294            else                      ii = nbyte - 1 - i;
1295
1296            chrwrk[ii] = string[i];
1297        }
1298        return( numb );
1299    }
1300
1301    float convfloat( unsigned char *string, int nbyte )
1302    {
1303        int                 i,ii;
1304        float               numb=0.;
1305        long                lbig;
1306        unsigned char       *chrwrk,*cbig;
1307
1308        lbig=1;
1309        cbig = (unsigned char *)&lbig;
1310
1311        chrwrk = (unsigned char *)&numb;
1312        for( i = 0; i < nbyte; i++ ) {
1313            if( 1 == cbig[3] & 0x01 ) ii = 4 - nbyte + i;
1314            else                      ii = nbyte - 1 - i;
1315
1316            chrwrk[ii] = string[i];
1317        }
1318        return( numb );
1319    }
1320
```

```
1321    long convlatlon(unsigned char *string )
1322    {
1323        long      ll;
1324        char      c,a;
1325        unsigned char wk[5];
1326
1327        c = MASK1 & string[0];
1328        if( 0 != c ){
1329                a = -1;}
1330            else{
1331                a = 1;
1332            }
1333
1334        wk[0] = MASK7 & string[0];
1335        wk[1]=string[1];wk[2]=string[2];wk[3]=string[3];
1336        ll = (long)a*convlong(wk,(int)4);
1337        return(ll);
1338    }
1339
1340    char mem2char(char *string)
1341    {
1342        int rt;
1343              rt = string[0];
1344        if(( rt & MASK1 ) && rt != -1 ) rt = -(rt & MASK7);
1345              return rt;
1346    }
1347
1348    /*=========================================================================*/
1349    /*      used in identifydata                                               */
1350    /*=========================================================================*/
1351    void fproduct(void)
1352        {
1353        char      wk[4];
1354        fl[0]='X';
1355        fl[1]='X';
1356        fl[2]='X';
1357        if( mtn == 0)
1358          {
1359             if( pc == 0 && pn == 0 ){fl[0]='T';fl[1]='_';fl[2]='_';}
1360           if( pc == 0 && pn == 49){fl[0]='P';fl[1]='W';fl[2]='G';}
1361           if( pc == 0 && pn == 9 ){fl[0]='T';fl[1]='A';fl[2]='_';}
1362           if( pc == 1 && pn == 0 ){fl[0]='Q';fl[1]='Q';fl[2]='_';}
1363           if( pc == 1 && pn == 1 ){fl[0]='R';fl[1]='H';fl[2]='_';}
1364           if( pc == 1 && pn == 8 ){fl[0]='R';fl[1]='R';fl[2]='R';}
1365           if( pc == 1 && pn ==20 ){fl[0]='r';fl[1]='r';fl[2]='r';}
1366           if( pc == 1 && pn ==210){fl[0]='R';fl[1]='R';fl[2]='d';}
1367           if( pc == 1 && pn ==211){fl[0]='R';fl[1]='A';fl[2]='d';}
1368           if( pc == 1 && pn ==212){fl[0]='Q';fl[1]='Q';fl[2]='A';}
1369           if( pc == 1 && pn ==213){fl[0]='R';fl[1]='H';fl[2]='A';}
1370           if( pc == 2 && pn == 1 ){fl[0]='W';fl[1]='_';fl[2]='_';}
1371           if( pc == 2 && pn == 2 ){fl[0]='U';fl[1]='_';fl[2]='_';}
1372           if( pc == 2 && pn == 3 ){fl[0]='V';fl[1]='_';fl[2]='_';}
1373           if( pc == 2 && pn == 4 ){fl[0]='P';fl[1]='S';fl[2]='I';}
```

```
1374            if( pc == 2 && pn == 5 ){fl[0]='C';fl[1]='H';fl[2]='I';}
1375            if( pc == 2 && pn ==210){fl[0]='U';fl[1]='A';fl[2]='_';}
1376            if( pc == 2 && pn ==211){fl[0]='V';fl[1]='A';fl[2]='_';}
1377            if( pc == 3 && pn == 0 ){fl[0]='P';fl[1]='_';fl[2]='_';}
1378            if( pc == 3 && pn == 1 ){fl[0]='P';fl[1]='s';fl[2]='a';}
1379            if( pc == 3 && pn == 4 ){fl[0]='z';fl[1]='_';fl[2]='_';}
1380            if( pc == 3 && pn == 5 ){fl[0]='Z';fl[1]='_';fl[2]='_';}
1381            if( pc == 3 && pn == 8 ){fl[0]='P';fl[1]='A';fl[2]='_';}
1382            if( pc == 3 && pn == 9 ){fl[0]='Z';fl[1]='A';fl[2]='_';}
1383            }
1384          else if(mtn == 10 )
1385            {
1386            if( pc == 3 && pn == 0 ){fl[0]='S';fl[1]='T';fl[2]='_';}
1387            if( pc == 3 && pn ==192){fl[0]='S';fl[1]='T';fl[2]='A';}
1388            }
1389          fl[3]='?';
1390          fl[4]='?';
1391          fl[5]='?';
1392          if( nmem <     0 ){fl[3]='_';fl[4]='_';fl[5]='_';}
1393          if( nmem >=    0 && nmem < 100 ){
1394              switch (tyens)
1395                {
1396                case      1:
1397                   fl[5] = '_'; break;
1398                case      2:
1399                   fl[5] = 'm'; break;
1400                case      3:
1401                   fl[5] = 'p'; break;
1402                default:
1403                   printf(" tyens Not Supported !! tyens=%d ¥n",tyens);exit(71);
1404                }
1405              sprintf(wk,"%02d",nmem);fl[3]=wk[0];fl[4]=wk[1];
1406             }
1407          if( nmem >= 100 ){printf(" Over 100 members!! Not Supported.¥n");exit(70);}
1408
1409          }
1410
1411    void flevel(void)
1412          {
1413          char       wk[10];
1414          unsigned long vl1hpa;
1415          vl1hpa=vl1*0.01;
1416          fl[6]='X';
1417          fl[7]='X';
1418          fl[8]='X';
1419          fl[9]='X';
1420          if( tys == 1 ){fl[6]='_';fl[7]='S';fl[8]='F';fl[9]='C';}
1421          if( tys == 100 ){sprintf(wk,"%04d",vl1hpa);
1422                fl[6]=wk[0];fl[7]=wk[1];fl[8]=wk[2];fl[9]=wk[3];}
1423          if( tys == 101 ){fl[6]='_';fl[7]='S';fl[8]='E';fl[9]='A';}
1424          if( tys == 103 ){fl[6]='_';fl[7]='2';fl[8]='M';fl[9]='_';}
1425
1426          }
```

```
1427    void fderived(void)
1428        {
1429        fl[10]='X';fl[11]='X';fl[12]='X';
1430        if( dfn == 0 ){fl[10]='E';fl[11]='S';fl[12]='_';}/* Unweighted mean of all members
1431         */
1432        if( dfn == 1 ){fl[10]='E';fl[11]='S';fl[12]='W';}/* Weighted mean of all members
1433         */
1434        if( dfn == 4 ){fl[10]='S';fl[11]='P';fl[12]='R';}/* Spread of all members
1435         */
1436        if( dfn == 5 ){fl[10]='L';fl[11]='A';fl[12]='I';}/* Large Anomaly Index of all members
1437         */
1438        if( dfn == 6 ){fl[10]='C';fl[11]='L';fl[12]='S';}/* Unweighted mean of the cluster members */
1439        if( Npdt == 11 ){fl[10]='M';fl[11]='E';fl[12]='M';}/* All member */
1440        }
1441    void fhemispher(void)
1442        {
1443        fl[13]='?';fl[14]='?';
1444        if(La1 >= 80000000 && La2 <= -80000000 ){fl[13]='-';fl[14]='G';}
1445        if(La1 == 90000000 && La2 == 0 ) {fl[13]='-';fl[14]='N';}
1446        if(La1 == 0            && La2 == -90000000 ){fl[13]='-';fl[14]='S';}
1447        if(La1 == 90000000 && La2 == -90000000 ){fl[13]='-';fl[14]='G';}
1448
1449        }
1450    void fperiod(void)
1451        {
1452        fl[15]='.';fl[16]='F';fl[17]='?';fl[18]='?';fl[19]='?';fl[20]='E';
1453        fl[21]='?';fl[22]='?';fl[23]='?';fl[24]='?';fl[25]='?';
1454        sprintf(&fl[15],".F%03dE%03d",kt,lt);
1455
1456        }
1457
1458    /*=========================================================================*/
1459    /*     Identify Data (FILE NAME)                                           */
1460    /*=========================================================================*/
1461    void identifydata(void)
1462        {
1463        int       i;
1464        for( i = 0 ; i < 31 ; ++i){
1465             fl[i]=' ';
1466             }
1467        fproduct();
1468        flevel();
1469        fderived();
1470        fhemispher();
1471        fperiod();
1472        fl[26]=0x00;
1473        /*printf(" FNAME= %s¥n¥n",fl);*/
1474        }
1475
```

1か月予報アンサンブル統計全球格子点値ファイルにおけるGRIB2第４節の補足説明

１．GRIB2のフォーマット及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4節	プロダクト定義節	1～4	節の長さ		***	
		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	12	連続または不連続な時間間隔の水平面または水平層における全てのアンサンブルメンバーを用いたデライブド予報
	ここから ﾃﾝﾌﾟﾚｰﾄ4.12	10	パラメータカテゴリー	符号表4．1	*	
	↓	11	パラメータ番号	符号表4．2	*	
	↓	12	作成処理の種類	符号表4．3	4	アンサンブル予報
	↓	13	背景作成処理識別符	**JMA定義**	102	アンサンブル1か月予報モデル
	↓	14	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）		2	
	↓	17	観測資料の参照時刻からの締切時間（分）		30	
	↓	18	期間の単位の指示符	符号表4．4	2	日
	↓	19～22	予報時間-単位は第18オクテットで定義		****	予報時間
	↓	23	第一固定面の種類	符号表4．5	*	
	↓	24	第一固定面の尺度因子		*	
	↓	25～28	第一固定面の尺度付きの値		*	
	↓	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		*	
	↓	31～34	第二固定面の尺度付きの値		*	
	↓	35	デライブド予報	符号表4．7	*	全メンバーによる非加重平均(0、スプレッド(4)、高偏差(5)
	↓	36	アンサンブルにおける予報の数		*	アンサンブルメンバー数
	↓	37～38	全時間間隔の終了時（年）		*	
	↓	39	全時間間隔の終了時（月）		*	
	↓	40	全時間間隔の終了時（日）		*	
	↓	41	全時間間隔の終了時（時）		*	
	↓	42	全時間間隔の終了時（分）		*	
	↓	43	全時間間隔の終了時（秒）		*	
	↓	44	統計を算出するために使用した時間間隔を記述する期間の仕様の数		1	
	↓	45～48	統計処理における欠測資料の総数		*	
	↓	49	統計処理の種類	符号表4．10	0	平均
	↓	50	統計処理の時間増分の種類	符号表4．11	2	同じ予報開始時刻を持ち、予報時間に増分が加えられる
	↓	51	統計処理の時間の単位の指示符	符号表4．4	*	11:6時間 2:日
	↓	52～55	統計処理した期間の長さ		****	平均回数
	↓	56	連続的な資料場間の増分に関する時間の単位の指示符	符号表4．4	*	11:6時間 2:日
	↓	57～60	連続的な資料場間の時間の増分		*	統計処理に利用したデータの時間間隔（1日各のデータを1か月平均した場合は1日）、連続量の場合は0

1か月予報メンバー別全球格子点値ファイルにおけるGRIB2第4節の補足説明

1．GRIB2のフォーマット及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4節	プロダクト定義節	1～4	節の長さ		*	
		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	11	連続または不連続な時間間隔の水平面または水平層における個々のアンサンブル予報
	ここから テンプレート4.11	10	パラメータカテゴリー	符号表4．1	*	
	↓	11	パラメータ番号	符号表4．2	*	
	↓	12	作成処理の種類	符号表4．3	4	アンサンブル予報
	↓	13	背景作成処理識別符	**JMA定義**	102	アンサンブル1か月予報モデル
	↓	14	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）		2	
	↓	17	観測資料の参照時刻からの締切時間（分）		30	
	↓	18	期間の単位の指示符	符号表4．4	2	日
	↓	19～22	予報時間		***	予報時間
	↓	23	第一固定面の種類	符号表4．5	*	
	↓	24	第一固定面の尺度因子		*	
	↓	25～28	第一固定面の尺度付きの値		*	
	↓	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		*	
	↓	31～34	第二固定面の尺度付きの値		*	
	↓	35	アンサンブル予報の種類	符号表4．6	*	1：コントロール、2：負の摂動、3：正の摂動
	↓	36	摂動番号		*	
	↓	37	アンサンブルにおける予報の数		*	アンサンブルメンバー数
	↓	38～39	全時間間隔の終了時（年）		**	
	↓	40	全時間間隔の終了時（月）		*	
	↓	41	全時間間隔の終了時（日）		*	
	↓	42	全時間間隔の終了時（時）		*	
	↓	43	全時間間隔の終了時（分）		*	
	↓	44	全時間間隔の終了時（秒）		*	
	↓	45	統計を算出するために使用した時間間隔を記述する期間の仕様の数		1	
	↓	46～49	統計処理における欠測資料の総数		****	
	↓	50	当該期間中それぞれの時間増分における資料場から処理を算出するために用いた統計	符号表4．10	*	0：平均、1：積算
	↓	51	統計処理の時間増分の種類	符号表4．11	2	同じ予報開始時刻を持ち、予報時間に増分が加えられる
	↓	52	統計処理の時間の単位の指示符	符号表4．4	11	6時間
	↓	53～56	統計処理した期間の長さ		4	平均回数
	↓	57	連続的な資料場間の増分に関する時間の単位の指示符	符号表4．4	11	6時間
	↓	58～61	連続的な資料場間の時間の増分		*	統計処理に利用したデータの時間間隔（1日各のデータを1か月