平成１５年６月５日
気象庁気候・海洋気象部

配信資料に関する技術情報（気象編）第139号

～３か月予報ＧＰＶ等のサンプルデータの提供について～

配信資料に関する技術情報（気象編）第133号（平成15年5月12日発表）で、３か月予報ＧＰＶ等のFTP方式による提供開始及びそのフォーマットについてお知らせしました。これに関連し、本日、支援センターにサンプルファイルを提供しますのでお知らせします。

１．サンプルファイルを収録した２枚のCD-ROMの構成について

ディスク１

フォルダ名	格納ファイル
ＧＰＶ	アンサンブル統計格子点値（２９ファイル）及び同 tar 圧縮ファイル
GDC	ガイダンス（７ファイル）及び同 tar 圧縮ファイル
OCNCCA	統計予測資料（４ファイル）及び同 tar 圧縮ファイル
MGPVnh	メンバー別格子点値（北半球、１６ファイル）

ディスク２

フォルダ名	格納ファイル
MGPVsh	メンバー別格子点値（南半球、１６ファイル）

2. 解読（デコード）処理について

３か月予報アンサンブル格子点値（メンバー別格子点値及びアンサンブル統計格子点値）は解読処理が必要です。

添付資料別添１ではアンサンブル統計格子点値、添付資料別添２ではメンバー別格子点値についての、解読サンプルプログラムおよびその利用方法を解説しておりますので参考にしてください。

なお、３か月予報支援資料（ガイダンス及び統計予測資料）は、CSV形式（カンマで区切られたテキストデータ）のため、解読処理は必要ありません。

３か月予報アンサンブル統計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_3megrib2_sample.c
実行ファイル名　　：dcd_3merib2
とする。

（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_3megrib2_sample.c - lm - o dcd_3merib2
（dcd_3megrib2 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, HI-UX/WE2) 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、３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る。
実行例）$ dcd_3megrib2 ｛３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757行で使用されている関数 identifydata 104～224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５行目の
#define IFOUT ″yes″
を
#define IFOUT ″no”
など、yes以外の文字列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
（３か月予報アンサンブル統計格子点値ファイル用）

```
#include <stdio.h>
#include <string.h>
#include <math.h>

#define IFOUT "yes"                     /*   yes :Output decoded data                */
#define MASK7   0x7F
#define MASK1   0x80
#define UNDEF   -19999.
#define SIZEOFBITBUF 50000
#define MAX_BUFF        50000           /*   buffer size for gpv                     */

      static      char      fl[30];

/*=============================================================================*/
/* TOOLS                                                                       */
/*=============================================================================*/
usage()
{
          puts("¥n");
          puts("Usage : GRIB2 <FILE-NAME>");
}


long longbybit(buf,pos,nbit)
      unsigned long      *pos,nbit;
      unsigned char      buf[MAX_BUFF];
      /* MAX_BUFF < 4294967295   2^32 */
{
      char      i,ii;
      unsigned long      pos8,mg8,rem8,nbytes;
      long                lout=0;
      long                lbig;
      unsigned char      *buf2,*cbig;
      unsigned char      mask,full=0xff;

      lbig=1;
      cbig = (unsigned char *)&lbig;

      buf2 = (unsigned char *)&lout;
      pos8 = *pos / 8;
      mg8   = *pos % 8;

      if( (*pos + nbit) % 8 == 0 ) rem8 = 0;
      else                              rem8 = 8 - (( *pos + nbit ) % 8 );

      nbytes = (mg8 + nbit + rem8 ) / 8;
      mask    = full >>   mg8 + rem8;

      for( i = 0;i < nbytes -1 ;i++ ){
```

```
        if( 1 == cbig[3] & 0x01 ) ii = 3 -i;
        else                      ii = i     ;
        buf2[ii] = ( buf[pos8 + nbytes -1 - i] >> rem8
                   | buf[pos8 + nbytes -1 - i - 1] << 8 - rem8 );
    }
    if( 1 == cbig[3] & 0x01 ) ii = 4 - nbytes;
    else                      ii = nbytes -1;
    buf2[ii] = ( buf[pos8] >> rem8 & mask );

    *pos = *pos + nbit;
    return(lout);
}



long convlong( unsigned char *string, int nbyte )
{
    int             i,ii;
    long            numb=0;
    long            lbig;
    unsigned char   *chrwrk,*cbig;

    lbig=1;
    cbig = (unsigned char *)&lbig;
    /*if( 1 == cbig[3] & 0x01 ) printf("BIG ENDIAN !!");*/
    chrwrk = (unsigned char *)&numb;
    for( i = 0; i < nbyte; i++ ) {
        if( 1 == cbig[3] & 0x01 ) ii = 4 - nbyte + i;
        else                      ii = nbyte - 1 - i;

        chrwrk[ii] = string[i];
    }
    return( numb );
}

float convfloat( unsigned char *string, int nbyte )
{
    int             i,ii;
    float           numb=0.;
    long            lbig;
    unsigned char   *chrwrk,*cbig;

    lbig=1;
    cbig = (unsigned char *)&lbig;

    chrwrk = (unsigned char *)&numb;
    for( i = 0; i < nbyte; i++ ) {
        if( 1 == cbig[3] & 0x01 ) ii = 4 - nbyte + i;
        else                      ii = nbyte - 1 - i;

        chrwrk[ii] = string[i];
    }
    return( numb );
```

```
}
/*============================================================================*/
/*      used in identifydata()                                                */
/*============================================================================*/
void fproduct(fl,mtn,pc,pn,nmem)
      unsigned long       mtn,pc,pn;
      char      fl[30];
      long                nmem;
      {
      char      wk[4];
      fl[0]='X';
      fl[1]='X';
      fl[2]='X';
      if( mtn == 0)
        {
          if( pc == 0 && pn == 0 ){fl[0]='T';fl[1]='_';fl[2]='_';}
        if( pc == 0 && pn == 49){fl[0]='P';fl[1]='W';fl[2]='G';}
        if( pc == 0 && pn == 9 ){fl[0]='T';fl[1]='A';fl[2]='_';}
        if( pc == 0 && pn ==129){fl[0]='Z';fl[1]='_';fl[2]='_';}
        if( pc == 0 && pn ==130){fl[0]='T';fl[1]='_';fl[2]='_';}
        if( pc == 1 && pn == 1 ){fl[0]='R';fl[1]='H';fl[2]='_';}
        if( pc == 1 && pn == 8 ){fl[0]='R';fl[1]='R';fl[2]='R';}
        if( pc == 1 && pn ==20 ){fl[0]='r';fl[1]='r';fl[2]='r';}
        if( pc == 1 && pn ==210){fl[0]='R';fl[1]='R';fl[2]='d';}
        if( pc == 1 && pn ==211){fl[0]='R';fl[1]='A';fl[2]='d';}
        if( pc == 2 && pn == 1 ){fl[0]='W';fl[1]='_';fl[2]='_';}
        if( pc == 2 && pn == 2 ){fl[0]='U';fl[1]='_';fl[2]='_';}
        if( pc == 2 && pn == 3 ){fl[0]='V';fl[1]='_';fl[2]='_';}
        if( pc == 2 && pn == 4 ){fl[0]='P';fl[1]='S';fl[2]='I';}
        if( pc == 2 && pn == 5 ){fl[0]='C';fl[1]='H';fl[2]='I';}
        if( pc == 2 && pn ==210){fl[0]='U';fl[1]='A';fl[2]='_';}
        if( pc == 2 && pn ==211){fl[0]='V';fl[1]='A';fl[2]='_';}
        if( pc == 3 && pn == 0 ){fl[0]='P';fl[1]='_';fl[2]='_';}
        if( pc == 3 && pn == 1 ){fl[0]='P';fl[1]='s';fl[2]='a';}
        if( pc == 3 && pn == 4 ){fl[0]='z';fl[1]='_';fl[2]='_';}
        if( pc == 3 && pn == 5 ){fl[0]='Z';fl[1]='_';fl[2]='_';}
        if( pc == 3 && pn == 8 ){fl[0]='P';fl[1]='A';fl[2]='_';}
        if( pc == 3 && pn == 9 ){fl[0]='Z';fl[1]='A';fl[2]='_';}
        }
      else if(mtn == 10 )
        {
        if( pc == 3 && pn == 0 ){fl[0]='S';fl[1]='T';fl[2]='_';}
        if( pc == 3 && pn ==192){fl[0]='S';fl[1]='T';fl[2]='A';}
        }
      fl[3]='?';
      fl[4]='?';
      fl[5]='?';
      if( nmem <     0 ){fl[3]='_';fl[4]='_';fl[5]='_';}
      if( nmem >=    0 && nmem < 100 ){fl[3]='e';sprintf(wk,"%02d",nmem);fl[4]=wk[0];fl[5]=wk[1];}
      if( nmem >= 100 ){printf(" Over 100 members!! Not Supported.¥n");exit(70);}

      }
void flevel(fl,tys,vl1)
```

```
	unsigned long tys,vl1;
	char	fl[30];
	{
	char	wk[10];
	unsigned long vl1hpa;
	vl1hpa=vl1*0.01;
	fl[6]='X';
	fl[7]='X';
	fl[8]='X';
	if( tys == 1 ){fl[6]='S';fl[7]='F';fl[8]='C';}
	if( tys == 100 ){sprintf(wk,"%03d",vl1hpa);fl[6]=wk[0];fl[7]=wk[1];fl[8]=wk[2];}
	if( tys == 101 ){fl[6]='S';fl[7]='E';fl[8]='A';}
	if( tys == 103 ){fl[6]='2';fl[7]='M';fl[8]='_';}

	}
void fderived(fl,dfn)
	unsigned long dfn;
	char	fl[30];
	{
	fl[9]='X';fl[10]='X';fl[11]='X';
	if( dfn == 0 ){fl[9]='E';fl[10]='S';fl[11]='_';}/* Unweighted mean of all members             */
	if( dfn == 1 ){fl[9]='E';fl[10]='S';fl[11]='W';}/* Weighted mean of all members               */
	if( dfn == 4 ){fl[9]='S';fl[10]='P';fl[11]='R';}/* Spread of all members                      */
	if( dfn == 5 ){fl[9]='L';fl[10]='A';fl[11]='I';}/* Large Anomaly Index of all members         */
	if( dfn == 6 ){fl[9]='C';fl[10]='L';fl[11]='S';}/* Unweighted mean of the cluster members */

	}
void fhemispher(fl,La1,La2)
	unsigned long La1,La2;
	char	fl[30];
	{
	fl[12]='?';fl[13]='?';
	if(La1 >= 80000000 && La2 <= -80000000 ){fl[12]='-';fl[13]='G';}
	if(La1 == 90000000 && La2 == 0 ) {fl[12]='-';fl[13]='N';}
	if(La1 == 0              && La2 == -90000000 ){fl[12]='-';fl[13]='S';}
	if(La1 == 90000000 && La2 == -90000000 ){fl[12]='-';fl[13]='G';}

	}
void fperiod(fl,kt,lt)
	unsigned long kt,lt;
	char	fl[30];
	{
	fl[14]='.';fl[15]='F';fl[16]='?';fl[17]='?';fl[18]='E';fl[19]='?';
	fl[20]='?';fl[21]='?';fl[22]='?';fl[23]='?';fl[24]='?';
	sprintf(&fl[14],".F%03dE%03d",kt,lt);

	}

/*============================================================================*/
/*     Identify Data (FILE NAME)                                              */
/*============================================================================*/
void identifydata(fl,La1,La2,mtn,pc,pn,tys,sfl1,vl1,dfn,kt,lt,nmem)
	unsigned long La1,La2,mtn,pc,pn,tys,vl1,dfn,kt,lt;
```

```
        long       nmem;
        char                sfl1;
        char       fl[30];
        {
        int        i;
        for( i = 0 ; i < 30 ; ++i){
              fl[i]=' ';
              }
        fproduct(fl,mtn,pc,pn,nmem);
        flevel(fl,tys,vl1);
        fderived(fl,dfn);

        fhemispher(fl,La1,La2);
        fperiod(fl,kt,lt);
        fl[25]=0x00;
        }
int month2str(mon,imm)
        char       mon[4];
        unsigned long imm;
        {
        char *MO;
        MO=mon;
        switch(imm)
              {
              case   1: memmove(MO,"jan",3);break;
              case   2: memmove(MO,"feb",3);break;
              case   3: memmove(MO,"mar",3);break;
              case   4: memmove(MO,"apr",3);break;
              case   5: memmove(MO,"may",3);break;
              case   6: memmove(MO,"jun",3);break;
              case   7: memmove(MO,"jul",3);break;
              case   8: memmove(MO,"aug",3);break;
              case   9: memmove(MO,"sep",3);break;
              case 10: memmove(MO,"oct",3);break;
              case 11: memmove(MO,"nov",3);break;
              case 12: memmove(MO,"dec",3);break;
              default:printf(" Unkown month imm=%d¥n",imm);return(99);
              }
              return(0);
        }
long convlatlon(unsigned char *string )
{
        long       ll;
        char       c,a;
        unsigned char wk[5];

        c = MASK1 & string[0];
        if( 0 != c ){
                    a = -1;}
              else{
                    a = 1;
              }
```

```
	wk[0] = MASK7 & string[0];
	wk[1]=string[1];wk[2]=string[2];wk[3]=string[3];
	ll = (long)a*convlong(wk,(int)4);
	return(ll);
}
/*============================================================================*/
/* MAIN PROGRAM                                                               */
/*============================================================================*/
main(argc,argv)
int	argc;
char	**argv;
{
	int	i,j,k,l,recno;
	int		ii,ii1,ii2,itot;
	double	retbybit;
	FILE	*fp,*fpg,*fpc;
	unsigned long	len=0,mtn=0;
	unsigned char	buffer[200000],*buff,wk[4],map[SIZEOFBITBUF],buffer2[MAX_BUFF];
	unsigned long	len1=0,ns1=0,idcenter=0,icsubc=0,iyyyy=0,imm=0,iday=0;
	unsigned long	len2=0,ns2=0;
	unsigned long	len3=0,ns3=0,ijmx=0,Ngdt=0,wkl=0;
	unsigned long	Ni=0,Nj=0,Di=0,Dj=0,Ntoe=0,np[1000],mxnp,ol;
	unsigned long	La1=0,Lo1=0,La2=0,Lo2=0,la2;
	unsigned long	len4=0,ns4=0,noc=0,Npdt=0,tcut=0;
	unsigned long	kt=0,svl1=0,svl2=0,pc=0,pn=0,tys=0,vl1=0,dfn=0;
	long		nmem;
	unsigned long	iyyyye=0,imme=0,idaye=0,Nt=0,nmiss=0,lt=0,tinc=0;
	unsigned long	len5=0,ns5=0,ijdim=0,ntemplate=0,nbit=0;
	char		sfl1=0,ib;
	char		flg[20],mon[4];
	long		E=0,D=0;
	float		di,dj;
	float		R=0,fnbit,pos;
	unsigned long	*posi,lgpv;
	unsigned char	*cbig;
	long		lbig;
	float		data[500000],wdata[500000];
	float		Ld[1000],Lw[1000],wei1,wei2;
	unsigned long	len6=0,ns6=0,ifbitmap=0;
	unsigned long	len7=0,ns7=0,nb,mb,nbitp;
	unsigned char	msk_bit=MASK1;
	unsigned long	latn=0,lats=0,lone=0,lonw=0,Nc=0,vstd=0,vdist=0;
	if(2 != argc)
		{
		usage();
		exit(0);
		}
	if(NULL == (fp = fopen(argv[1],"rb")))
		{
		printf("¥nCannot open FILE : %s¥n",argv[1]);
		usage();
		exit(1);
		}
```

```
        recno=0;
/*=============================================================================*/
/*     DECODE SECT 0                                                           */
/*=============================================================================*/
    /*   Check Header   */
        if( fread(buffer,sizeof(char), 4, fp) == 0 ) exit(2);
        while( buffer[0] != 'G' || buffer[1] != 'R' ||
               buffer[2] != 'I' || buffer[3] != 'B' ){
                printf("Cannot Find GRIB header !!¥n");

                fclose(fp);return 99;
                }
        printf("-----------------------¥n");
        printf("---- GRIB FILE !! -----¥n");
        printf("-----------------------¥n");
    fread(buffer,sizeof(char), 12, fp);
    mtn=convlong(&buffer[2], 1);
    printf("¥n<<SECTION 0>> : Indicator Section¥n");
    printf("GRIB Master Table Number : %d¥n",mtn);
    printf("GRIB Edition Number          : %d¥n",buffer[3]);
/*    if(convlong(&buffer[4], 4) == 0 ){*/
        printf("buffer[4]= %d¥n",convlong(&buffer[4],4));
    if( 0 == 0){
        len = convlong(&buffer[8], 4);
        printf("Total length of GRIB        : %d¥n",len);
        }
    else
        {
        printf("Larg Record !! Not Supported !!¥n¥n");
        exit(3);
        }
/*=============================================================================*/
/*     DECODE SECT 1                                                           */
/*=============================================================================*/
    printf("¥n<<SECTION 1>> : Identification Section ¥n");
    fread(buffer,sizeof(char), 5,fp);
    len1=convlong(&buffer[0], 4);
    ns1=convlong(&buffer[4], 1);
    fread(buffer,sizeof(char),len1-5,fp);
    idcenter=convlong(&buffer[0], 2);
    icsubc=convlong(&buffer[2],2);
    iyyyy=convlong(&buffer[7], 2);
    imm=convlong(&buffer[9], 1);
    iday=convlong(&buffer[10], 1);
    printf("Length of Section 1   : %d¥n",len1);
    printf("Number of Section      : %d¥n",ns1);
    printf("Identification of centre        : %d¥n",idcenter);
    printf("Identification of sub-centre : %d¥n",icsubc);
    printf("GRIB Master Tables Version Number : %d¥n",buffer[4]);
    printf("GRIB Local Tables Version Number   : %d¥n",buffer[5]);
    printf("Significance of Reference Time      : %d¥n",buffer[6]);
    printf("Year                           : %d¥n",iyyyy);
```

```
        printf("Month                          : %d¥n",imm);
        printf("Day                            : %d¥n",iday);
        printf("Hour                           : %d¥n",buffer[11]);
        printf("Minute                         : %d¥n",buffer[12]);
        printf("Second                         : %d¥n",buffer[13]);
        printf("Production status of processed data : %d¥n",buffer[14]);
        printf("Type of processed data                 : %d¥n",buffer[15]);

        printf("-----------------------¥n");
        printf("Record No=%d ¥n",++recno);
        printf("-----------------------¥n");

/*=============================================================================*/
/*     DECODE SECT 2                                                           */
/*=============================================================================*/
        if(buffer[5] != 0 ){
        printf("¥n<<SECTION 2>> : (Local Use Section)¥n");
        fread(buffer,sizeof(char), 5,fp);
        wkl=convlong(&buffer[0], 4);
SECT2:
        len2=wkl;
        ns2=convlong(&buffer[4], 1);
        printf("Length of Section 2    : %d¥n",len2);
        printf("Number of Section      : %d¥n",ns2);
        if( len2-4 != 0 ){
              fread(buffer,sizeof(char),len2-5,fp);
        printf(" Local use .... Not Supported!!");
        }
        }

/*=============================================================================*/
/*     DECODE SECT 3                                                           */
/*=============================================================================*/
        printf("¥n<<SECTION 3>> : Grid Definition Section¥n");
        if( fread(buffer,sizeof(char), 4,fp) != 4 ){
        printf("Read ERR SECT 3-1 !! File(%s)¥n",argv[1]);
        exit(74);
        }
        wkl=convlong(&buffer[0], 4);
        if( fread(buffer,sizeof(char), 1,fp) != 1 ){
        printf("Read ERR SECT 3-2 !! File(%s)¥n",argv[1]);
        exit(75);
        }
SECT3:
        len3=wkl;
        ns3=convlong(&buffer[0], 1);
        fread(buffer,sizeof(char),len3-5,fp);
        ijmx = convlong(&buffer[1], 4);
        Ngdt = convlong(&buffer[7], 2);
        ol     = convlong(&buffer[5], 1);
        printf("Length of section                 : %d¥n",len3);
        printf("Number of section                 : %d¥n",ns3);
        printf("Source of grid definition         : %d¥n",buffer[0]);
```

```
printf("Number of data points                  : %d¥n",ijmx);
printf("optional list                          : %d¥n",ol);
printf("Interpretation of list                 : %d¥n",buffer[6]);
printf("Grid Definition Template Number : %d¥n",Ngdt);
switch(Ngdt){
    case    0:
        printf("((Grid Definition Template 3.0))¥n");
        printf("Shape of the earth                                  : %d¥n",buffer[9]);
        printf("Scale factor of radius of spherical earth           : %d¥n",buffer[10]);
        wkl = convlong(&buffer[11], 4);
        printf("Scaled value of radius of spherical earth           : %d¥n",wkl);wkl=0;
        printf("Scale factor of major axis of oblate spheroid earth : %d¥n",buffer[15]);
        wkl =   convlong(&buffer[16], 4);
        printf("Scaled value of major axis of oblate spheroid earth : %d¥n",wkl);wkl=0;
        printf("Scale factor of minor axis of oblate spheroid earth : %d¥n",buffer[20]);
        wkl = convlong(&buffer[21], 4);
        printf("Scaled value of minor axis of oblate spheroid earth : %d¥n",wkl);wkl=0;
        Ni   = convlong(&buffer[25], 4);
        Nj   = convlong(&buffer[29], 4);
        printf("number of points along a parallel                   : %d¥n",Ni);
        printf("number of points along a meridian                   : %d¥n",Nj);
        wkl = convlong(&buffer[33], 4);
        printf("Basic angle of the initial production domain        : %d¥n",wkl);wkl=0;
        wkl = convlong(&buffer[37], 4);
        printf("Subdivisions of basic angle                         : %d¥n",wkl);wkl=0;
        La1 = convlong(&buffer[41], 4);
        Lo1 = convlong(&buffer[45], 4);
        printf("latitude of first grid point(La1)                   : %d¥n",La1);
        printf("longitude of first grid point(Lo1)                  : %d¥n",Lo1);
        printf("Resolution and component flags                      : %d¥n",buffer[49]);
        La2 = convlatlon(&buffer[50]);
        Lo2 = convlatlon(&buffer[54]);
        printf("latitude of last grid point(La2)                    : %d¥n",La2);
        printf("longitude of last grid point(Lo2)                   : %d¥n",Lo2);
        Di   = convlong(&buffer[58], 4);
        Dj   = convlong(&buffer[62], 4);
        printf("i direction increment(Di)                           : %d¥n",Di);
        printf("j direction increment(Dj)                           : %d¥n",Dj);
        printf("Scanning mode                                       : %d¥n",buffer[66]);
        break;
    case 40:
        printf("((Grid Definition Template 3.40))¥n");
        printf("Shape of the earth                                  : %d¥n",buffer[9]);
        printf("Scale factor of radius of spherical earth           : %d¥n",buffer[10]);
        wkl = convlong(&buffer[11], 4);
        printf("Scaled value of radius of spherical earth           : %d¥n",wkl);wkl=0;
        printf("Scale factor of major axis of oblate spheroid earth : %d¥n",buffer[15]);
        wkl =   convlong(&buffer[16], 4);
        printf("Scaled value of major axis of oblate spheroid earth : %d¥n",wkl);wkl=0;
        printf("Scale factor of minor axis of oblate spheroid earth : %d¥n",buffer[20]);
        wkl = convlong(&buffer[21], 4);
        printf("Scaled value of minor axis of oblate spheroid earth : %d¥n",wkl);wkl=0;
        Ni   = convlong(&buffer[25], 4);
```

```
            Nj    = convlong(&buffer[29], 4);
            printf("number of points along a parallel                    : %d¥n",Ni);
            printf("number of points along a meridian                    : %d¥n",Nj);
            wkl = convlong(&buffer[33], 4);
            printf("Basic angle of the initial production domain         : %d¥n",wkl);wkl=0;
            wkl = convlong(&buffer[37], 4);
            printf("Subdivisions of basic angle                          : %d¥n",wkl);wkl=0;
            La1 = convlong(&buffer[41], 4);
            Lo1 = convlong(&buffer[45], 4);
            printf("latitude of first grid point(La1)                    : %d¥n",La1);
            printf("longitude of first grid point(Lo1)                   : %d¥n",Lo1);
            printf("Resolution and component flags                       : %d¥n",buffer[49]);
            La2 = convlatlon(&buffer[50]);
            Lo2 = convlatlon(&buffer[54]);
            printf("latitude of last grid point(La2)                     : %d¥n",La2);
            printf("longitude of last grid point(Lo2)                    : %d¥n",Lo2);
            Di    = convlong(&buffer[58], 4);
            Ntoe= convlong(&buffer[62], 4);
            printf("i direction increment(Di)                            : %d¥n",Di);
            printf("Number of parallels from pole to equator(N)          : %d¥n",Ntoe);
            printf("Scanning mode                                        : %d¥n",buffer[66]);
            printf("Number of points along each grid line ¥n");
            printf("        Line       Points¥n");
            mxnp=0;
            for(i=0 ;i < Nj ;++i)
                {
                np[i]=convlong(&buffer[67+i*2], ol );
                if(np[i] > mxnp) mxnp=np[i];
                }
            break;
        default      :
            printf("This Grid is Not Supported !!");
            exit(10);
    }

/*=============================================================================*/
/*     DECODE SECT 4                                                           */
/*=============================================================================*/
    printf("¥n<<SECTION 4>> : Product Definition Section¥n");
    if( fread(buffer,sizeof(char), 4,fp) != 4 ){
    printf("Read ERR SECT 4-1 !! File(%s)¥n",argv[1]);
    exit(77);
    }
    wkl=convlong(&buffer[0], 4);
    if( fread(buffer,sizeof(char), 1,fp) != 1 ){
    printf("Read ERR SECT 4-2 !! File(%s)¥n",argv[1]);
    exit(78);
    }
SECT4:
    len4=wkl;
    ns4=convlong(&buffer[0], 1);
    fread(buffer,sizeof(char),len4-5,fp);
    noc = convlong(&buffer[0], 2);
```

```
Npdt= convlong(&buffer[2], 2);
printf("Length of section 4 : %d¥n",len4);
printf("Number of section     : %d¥n",ns4);
printf("Number of coordinates values after Template   : %d¥n",noc);
printf("Product Definition Template Number                : %d¥n",Npdt);
printf("(( Product Definition Template 4.%d ))¥n",Npdt);
switch(Npdt){
    case       2:
        pc       = convlong(&buffer[4], 1);
        pn       = convlong(&buffer[5], 1);
        tcut     = convlong(&buffer[9], 2);
        kt       = convlong(&buffer[13], 4);
        tys      = convlong(&buffer[17], 1);
        sfl1     = buffer[18];
        svl1     = convlong(&buffer[19], 4);
        vl1      = svl1*pow(10.0,(double)sfl1);
        svl2     = convlong(&buffer[25], 4);
        dfn      = convlong(&buffer[29], 1);
        nmem     = -1;
        printf("Parameter category                                : %d¥n",pc);
        printf("Parameter number                                  : %d¥n",pn);
        printf("Type of generating process                        : %d¥n",buffer[6]);
        printf("Background generating process identifier          : %d¥n",buffer[7]);
        printf("Forecast generating process identifier            : %d¥n",buffer[8]);
        printf("Hours after reference time of data cut-off        : %d¥n",tcut);
        printf("Minutes after reference time of data cut-off      : %d¥n",buffer[11]);
        printf("Indicator of unit of time range                   : %d¥n",buffer[12]);
        printf("Forecast time in units defined by octet 18        : %d¥n",kt);
        printf("Type of first fixed surface                       : %d¥n",tys);
        printf("Scale factor of first fixed surface               : %d¥n",sfl1);
        printf("Scaled value of first fixed surface               : %d¥n",svl1);
        printf("Type of second fixed surface                      : %d¥n",buffer[23]);
        printf("Scale factor of second fixed surface              : %d¥n",buffer[24]);
        printf("Scaled value of second fixed surface              : %d¥n",svl2);
        printf(" Derived forecast (see Code Table 4.7)            : %d¥n",dfn);
        printf("Number of forecasts in ensemble                   : %d¥n",buffer[30]);
        break;
    case      11:
        pc       = convlong(&buffer[4], 1);
        pn       = convlong(&buffer[5], 1);
        tcut     = convlong(&buffer[9], 2);
        kt       = convlong(&buffer[13], 4);
        tys      = convlong(&buffer[17], 1);
        sfl1     = buffer[18];
        svl1     = convlong(&buffer[19], 4);
        vl1      = svl1*pow(10.0,(double)sfl1);
        svl2     = convlong(&buffer[25], 4);
        nmem     = convlong(&buffer[30], 1);
        iyyyye = convlong(&buffer[32], 2);
        imme     = convlong(&buffer[34], 1);
        idaye    = convlong(&buffer[35], 1);
        Nt       = convlong(&buffer[39], 1);
        nmiss    = convlong(&buffer[40], 4);
```

```
        printf("Parameter category                                : %d¥n",pc);
        printf("Parameter number                                  : %d¥n",pn);
        printf("Type of generating process                        : %d¥n",buffer[6]);
        printf("Background generating process identifier          : %d¥n",buffer[7]);
        printf("Forecast generating process identifier            : %d¥n",buffer[8]);
        printf("Hours after reference time of data cut-off        : %d¥n",tcut);
        printf("Minutes after reference time of data cut-off      : %d¥n",buffer[11]);
        printf("Indicator of unit of time range                   : %d¥n",buffer[12]);
        printf("Forecast time in units defined by octet 18        : %d¥n",kt);
        printf("Type of first fixed surface                       : %d¥n",tys);
        printf("Scale factor of first fixed surface               : %d¥n",sfl1);
        printf("Scaled value of first fixed surface               : %d¥n",svl1);
        printf("Type of second fixed surface                      : %d¥n",buffer[23]);
        printf("Scale factor of second fixed surface              : %d¥n",buffer[24]);
        printf("Scaled value of second fixed surface              : %d¥n",svl2);
        printf("Type of ensemble forecast                         : %d¥n",buffer[29]);
        printf("Perturbation number                               : %d¥n",buffer[30]);
        printf("Number of forecasts in ensemble                   : %d¥n",buffer[31]);
        printf("Year                                : %d¥n",iyyyye);
        printf("Month                               : %d¥n",imme);
        printf("Day                                 : %d¥n",idaye);
        printf("Hour                                : %d¥n",buffer[36]);
        printf("Minut                               : %d¥n",buffer[37]);
        printf("Second                              : %d¥n",buffer[38]);
        printf("n - Number of time range            : %d¥n",Nt);
        printf("Total number of data values missing               : %d¥n",nmiss);
        for(i=0 ; i < Nt ; ++i){
            lt      = convlong(&buffer[47+i*12], 4);
            tinc    = convlong(&buffer[52+i*12], 4);
            printf("Statistical process used to calculate ....    : %d¥n",buffer[44+i*12]);
            printf("Type of time increment                            : %d¥n",buffer[45+i*12]);
            printf("Indicator of unit of time for time range          : %d¥n",buffer[46+i*12]);
            printf("Length of the time range                          : %d¥n",lt);
            printf("Indicator of unit of time for the increment       : %d¥n",buffer[51+i*12]);
            printf("Time increment between successive fields          : %d¥n",tinc);tinc=0;
        }
        break;
    case    12:
        pc      = convlong(&buffer[4], 1);
        pn      = convlong(&buffer[5], 1);
        tcut    = convlong(&buffer[9], 2);
        kt      = convlong(&buffer[13], 4);
        tys     = convlong(&buffer[17], 1);
        sfl1    = buffer[18];
        svl1    = convlong(&buffer[19], 4);
        vl1     = svl1*pow(10.0,(double)sfl1);
        svl2    = convlong(&buffer[25], 4);
        dfn     = convlong(&buffer[29], 1);
        nmem    = -1;
        iyyyye = convlong(&buffer[31], 2);
        imme    = convlong(&buffer[33], 1);
        idaye   = convlong(&buffer[34], 1);
        Nt      = convlong(&buffer[38], 1);
```

```
        nmiss   = convlong(&buffer[39], 4);
        printf("Parameter category                                  : %d¥n",pc);
        printf("Parameter number                                    : %d¥n",pn);
        printf("Type of generating process                          : %d¥n",buffer[6]);
        printf("Background generating process identifier            : %d¥n",buffer[7]);
        printf("Forecast generating process identifier              : %d¥n",buffer[8]);
        printf("Hours after reference time of data cut-off          : %d¥n",tcut);
        printf("Minutes after reference time of data cut-off        : %d¥n",buffer[11]);
        printf("Indicator of unit of time range                     : %d¥n",buffer[12]);
        printf("Forecast time in units defined by octet 18          : %d¥n",kt);
        printf("Type of first fixed surface                         : %d¥n",tys);
        printf("Scale factor of first fixed surface                 : %d¥n",sfl1);
        printf("Scaled value of first fixed surface                 : %d¥n",svl1);
        printf("Type of second fixed surface                        : %d¥n",buffer[23]);
        printf("Scale factor of second fixed surface                : %d¥n",buffer[24]);
        printf("Scaled value of second fixed surface                : %d¥n",svl2);
        printf(" Derived forecast (see Code Table 4.7)              : %d¥n",dfn);
        printf("Number of forecasts in ensemble                     : %d¥n",buffer[30]);
        printf("Year                                : %d¥n",iyyyye);
        printf("Month                               : %d¥n",imme);
        printf("Day                                 : %d¥n",idaye);
        printf("Hour                                : %d¥n",buffer[35]);
        printf("Minut                               : %d¥n",buffer[36]);
        printf("Second                              : %d¥n",buffer[37]);
        printf("n - Number of time range            : %d¥n",Nt);
        printf("Total number of data values missing                 : %d¥n",nmiss);
        for(i=0 ; i < Nt ; ++i){
            lt      = convlong(&buffer[46+i*12], 4);
            tinc    = convlong(&buffer[51+i*12], 4);
            printf("Statistical process used to calculate ....      : %d¥n",buffer[43+i*12]);
            printf("Type of time increment                              : %d¥n",buffer[44+i*12]);
            printf("Indicator of unit of time for time range            : %d¥n",buffer[45+i*12]);
            printf("Length of the time range                            : %d¥n",lt);
            printf("Indicator of unit of time for the increment         : %d¥n",buffer[50+i*12]);
            printf("Time increment between successive fields            : %d¥n",tinc);tinc=0;
        }
        break;
    case    13:
        pc      = convlong(&buffer[4], 1);
        pn      = convlong(&buffer[5], 1);
        tcut    = convlong(&buffer[9], 2);
        kt      = convlong(&buffer[13], 4);
        tys     = convlong(&buffer[17], 1);
        sfl1    = buffer[18];
        svl1    = convlong(&buffer[19], 4);
        vl1     = svl1*pow(10.0,(double)sfl1);
        svl2    = convlong(&buffer[25], 4);
        dfn     = convlong(&buffer[29], 1);
        nmem    = -1;
        latn    = convlong(&buffer[36], 4);
        lats    = convlong(&buffer[40], 4);
        lone    = convlong(&buffer[44], 4);
        lonw    = convlong(&buffer[48], 4);
```

```
            Nc      = convlong(&buffer[52], 1);
            vstd    = convlong(&buffer[54], 4);
            vdist   = convlong(&buffer[59], 4);
            iyyyye = convlong(&buffer[63], 2);
            imme    = convlong(&buffer[65], 1);
            idaye   = convlong(&buffer[66], 1);
            Nt      = convlong(&buffer[70], 1);
            nmiss   = convlong(&buffer[71], 4);
            printf("Parameter category                                  : %d¥n",pc);
            printf("Parameter number                                    : %d¥n",pn);
            printf("Type of generating process                          : %d¥n",buffer[6]);
            printf("Background generating process identifier            : %d¥n",buffer[7]);
            printf("Forecast generating process identifier              : %d¥n",buffer[8]);
            printf("Hours after reference time of data cut-off          : %d¥n",tcut);
            printf("Minutes after reference time of data cut-off        : %d¥n",buffer[11]);
            printf("Indicator of unit of time range                     : %d¥n",buffer[12]);
            printf("Forecast time in units defined by octet 18          : %d¥n",kt);
            printf("Type of first fixed surface                         : %d¥n",tys);
            printf("Scale factor of first fixed surface                 : %d¥n",sfl1);
            printf("Scaled value of first fixed surface                 : %d¥n",svl1);
            printf("Type of second fixed surface                        : %d¥n",buffer[23]);
            printf("Scale factor of second fixed surface                : %d¥n",buffer[24]);
            printf("Scaled value of second fixed surface                : %d¥n",svl2);
            printf("Derived forecast (see Code Table 4.7)               : %d¥n",dfn);
            printf("Number of forecasts in the ensemble (N)             : %d¥n",buffer[30]);
            printf("Cluster identifier                                  : %d¥n",buffer[31]);
            printf("Number of cluster to which the high resolution control
belongs : %d¥n",buffer[32]);
            printf("Number of cluster to which the low resolution control
belongs  : %d¥n",buffer[33]);
            printf("Total number of clusters                            : %d¥n",buffer[34]);
            printf("Clustering method (see Code Table 4.8)              : %d¥n",buffer[35]);
            printf("Northern latitude of cluster domain                 : %d¥n",latn);
            printf("Southern latitude of cluster domain                 : %d¥n",lats);
            printf("Eastern longitude of cluster domain                 : %d¥n",lone);
            printf("Western longitude of cluster domain                 : %d¥n",lonw);
            printf("Number of forecasts in the cluster                  : %d¥n",Nc);
            printf("Scale factor of standard deviation in the cluster   : %d¥n",buffer[53]);
            printf("Scaled value of standard deviation in the cluster   : %d¥n",vstd);
            printf("Scale factor of distance of the cluster from ensemble mean  : %d¥n",buffer[58]);
            printf("Scaled value of distance of the cluster from ensemble mean  : %d¥n",vdist);
            printf("Year                                : %d¥n",iyyyye);
            printf("Month                               : %d¥n",imme);
            printf("Day                                 : %d¥n",idaye);
            printf("Hour                                : %d¥n",buffer[67]);
            printf("Minut                               : %d¥n",buffer[68]);
            printf("Second                              : %d¥n",buffer[69]);
            printf("n - Number of time range            : %d¥n",Nt);
            printf("Total number of data values missing                 : %d¥n",nmiss);
            for(i=0 ; i < Nt ; ++i){
                lt      = convlong(&buffer[78+i*12], 4);
                tinc    = convlong(&buffer[83+i*12], 4);
                printf("Statistical process used to calculate ....      : %d¥n",buffer[75+i*12]);
```

```
                printf("Type of time increment                              : %d¥n",buffer[76+i*12]);
                printf("Indicator of unit of time for time range          : %d¥n",buffer[77+i*12]);
                printf("Length of the time range                            : %d¥n",lt);
                printf("Indicator of unit of time for the increment       : %d¥n",buffer[82+i*12]);
                printf("Time increment between successive fields          : %d¥n",tinc);tinc=0;
            }
                printf("List of Nc ensemble forecast numbers :");
            for(i=0 ; i < Nc ; ++i){
                printf(" %2d",buffer[74+Nt*12+i+1]);
            }
                printf("¥n");
            break;
        default:
            printf("This Product Definition Template is Not Supported !!");
            exit(11);
    }
/* Identify Data(elm lvl date period */
/*      printf("Go identifydata(La1,La2,pc,pn,tys,sfl1,vl1,dfn)¥n");*/
    identifydata(fl,La1,La2,mtn,pc,pn,tys,sfl1,vl1,dfn,kt,lt,nmem);


/*=============================================================================*/
/*    DECODE SECT 5                                                            */
/*=============================================================================*/
    printf("¥n<<SECTION 5>> : Data Representation Section¥n");
    fread(buffer,sizeof(char), 5,fp);
    len5=convlong(&buffer[0], 4);
    ns5=convlong(&buffer[4], 1);
    fread(buffer,sizeof(char),len5-5,fp);
    ijdim = convlong(&buffer[0], 4);
    ntemplate = convlong(&buffer[4], 2);

    printf("Length of section in octets                         : %d¥n",len5);
    printf("Number of section                                   : %d¥n",ns5);
    printf("Number of data points                               : %d¥n",ijdim);
    printf("Data Representation Template Number                 : %d¥n",ntemplate);
    switch(ntemplate){
        case    0:
            R = convfloat(&buffer[6], 4);
            E = convlong(&buffer[10], 2);
                if(E > 32768) E = (E - 32768 )*(-1);
            D = convlong(&buffer[12], 2);
                if(D > 32768) D = (D - 32768 )*(-1);
            nbit = convlong(&buffer[14], 1);
            printf("Reference value (R)                             : %f¥n",R);
            printf("Binary scale factor (E)                         : %d¥n",E);
            printf("Decimal scale factor (D)                        : %d¥n",D);
            printf("Number of bits .....                            : %d¥n",nbit);
            printf("Type of original field values                   : %d¥n",buffer[15]);
            break;
            default:
                printf("This Data Representation Template is Not Supported!!");
                exit(12);
```

```
        }
/*=============================================================================*/
/*    DECODE SECT 6                                                            */
/*=============================================================================*/

    printf("¥n<<SECTION 6>> : Bit-map Section¥n");
    fread(buffer,sizeof(char), 5,fp);
    len6=convlong(&buffer[0], 4);
    ns6=convlong(&buffer[4], 1);
    fread(buffer,sizeof(char), 1,fp);
    ifbitmap = convlong(&buffer[0], 1);
    printf("Length of section in octets                    : %d¥n",len6);
    printf("Number of section                               : %d¥n",ns6);
    printf("Bit-map indicator                               : %d¥n",ifbitmap);
    for( i = 0 ; i < SIZEOFBITBUF ;++i){
        map[i]=0xFF;
        }
    switch(ifbitmap){
    case        0:
        if ( len6-6 != fread(map,sizeof(char),len6-6,fp)){
            printf("Read ERR SECT 6-1 !! File(%s)¥n",argv[1]);
            exit(100);
            }
        break;
        case    255:
            printf("No BITMAP !!¥n");
            break;
        default:
            printf("This Bit-map indicator is Not Supportted!!");
            exit(13);
    }
/*=============================================================================*/
/*    DECODE SECT 7                                                            */
/*=============================================================================*/
    printf("¥n<<SECTION 7>> : Data Section¥n");
    fread(buffer,sizeof(char), 5,fp);
    len7=convlong(&buffer[0], 4);
    ns7=convlong(&buffer[4], 1);
    printf("Length of section in octets                    : %d¥n",len7);
    printf("Number of section                               : %d¥n",ns7);
    fread(buffer2,sizeof(char),len7-5,fp);
    switch(ntemplate){
        case 0:
            lbig=1;
            cbig = (unsigned char *)&lbig;

            posi = 0;
            for( i = 0 ;i < ijmx ;i++ ){
                nb = i / 8;
                mb = i % 8;
                if( 0 ==    (map[nb] & ( msk_bit >> mb ))){
                    data[i]=UNDEF;
```

```
                    }
                else{
                    lgpv=longbybit(buffer2,&posi,nbit);
                    data[i] = (R + lgpv*pow(2.0,(double)E))/pow(10.0,(double)D);
                    }
                }

            printf("Check DATA !!       ¥n");
            printf("     No         value ¥n");
            for( i = 0; i < 10 ; ++i){
                printf("    %5d       %f¥n",i+1,data[i]);
            }
            printf("¥n¥n");
            for( i = ijmx-10; i < ijmx ; ++i){
                printf("    %5d       %f¥n",i+1,data[i]);
            }

            break;
        default:
            printf("Unsupported Template !!");
            exit(14);
    }
/*    OutPut Data    */
if( IFOUT == "yes" ){
    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 )
                {
                printf("Write ERR !! File(%s)¥n",fl);
                exit(98);
                }
            break;
        case 40:
            itot=0;
                for(i    = 0 ; i    < mxnp    ; ++i    )
                    {
                    Lw[i]=    360.0 * i    /    (mxnp);
                    }
            for(j    = 0 ; j    < Nj       ; ++j    )
                {
                for(ii = 0 ; ii < np[j]+1 ; ++ii )
                    {
                    Ld[ii]= 360.0 * ii /    (np[j]);
                    }
                  for(i    = 0 ; i    < mxnp    ; ++i    )
                    {
                    ii=0;ii1=0;ii2=0;
                    while( Lw[i] >= Ld[ii] )
```

```
                        {
                        if( Lw[i] == Ld[ii] )
                            {
                            ii1=ii ;ii2=ii;
                            }
                        if( Lw[i] >    Ld[ii] )
                            {
                            ii1=ii ;
                            ii2=ii+1;
                            }
                        ii=ii+1;
                        }
                    if( ii1 != ii2 )
                        {
                        wei1=( Ld[ii2] - Lw[i] )/(Ld[ii2] - Ld[ii1] );
                        wei2=1.0 - wei1;
                        }
                    else
                        {
                        wei1=0.5;
                        wei2=0.5;
                        }
                        wdata[j*mxnp+i] = data[ itot + ii1 ]*wei1 + data[ itot + ii2 ]*wei2;
                    }
                    itot = itot + np[j];
                }
                for( i = 0 ; i < Nj*mxnp ; ++i )
                    {
                    data[i] = wdata[i];
                    }
                Ni=mxnp;
                Di=360000000/(mxnp);
                Dj=(La1 - La2)/Nj;
                if( fwrite(data,4,Nj*mxnp,fpg) != Nj*mxnp )
                    {
                    printf("Write ERR !! File(%s)¥n",fl);
                    exit(96);
                    }
                break;
            default:
                printf(" This GDT is not sopported !!    Ngdt= %d ¥n",Ngdt);
                exit(95);
            }
        if ( fclose(fpg) != 0 ){
            printf("Close ERR !! File(%s)¥n",fl);
            exit(97);
            }
        }
/*=============================================================================*/
/*     DECODE SECT 8                                                           */
/*=============================================================================*/
```

```
    if( fread(buffer,sizeof(char), 4,fp) != 4 ){
    printf("Read ERR !! File(%s)¥n",argv[1]);
    exit(70);
    }
    if(buffer[0] == '7' && buffer[1] == '7' &&    buffer[2] == '7' && buffer[3] == '7'){
        printf("GRIB END !!");
        exit(0);
    }
    else{
        wkl=convlong(&buffer[0], 4);
    if( fread(buffer,sizeof(char), 1,fp) != 1 ){
        printf("Read ERR !! File(%s)¥n",argv[1]);
        exit(71);
    }
        switch(buffer[0]){
            case 2:
                printf("¥n----------¥n");
                printf("Record No=%d ¥n",++recno);
                printf("----------¥n");
                printf("<<SECTION 2>> : (Local Use Section)¥n");
                goto SECT2;
            case 3:
                printf("¥n----------¥n");
                printf("Record No=%d ¥n",++recno);
                printf("----------¥n");
                printf("¥n<<SECTION 3>> : Grid Definition Section¥n");
                goto SECT3;
            case 4:
                printf("¥n----------¥n");
                printf("Record No=%d ¥n",++recno);
                printf("----------¥n");
                printf("¥n<<SECTION 4>> : Product Definition Section¥n");
                goto SECT4;
            default:
                printf("ERROR in Section 8 !!");
                printf(" len ns = %d %d¥n",wkl,buffer[0]);
                exit(99);
        }
    }
}
```

３か月予報メンバー別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_3mepac_sample.f
実行ファイル名　　：dcd_3mepac
とする。

（１）FORRAN90コンパイラ(f90コマンドなど)によりコンパイルしてください。
実行例）　$ f90 dcd_3mepac_sample.f　- o dcd_3mepac
（dcd_3mepac と言う実行ファイルが生成される）

- JIS 標準 fortran90 コンパイラでコンパイルできます。日立最適化 fortran(HP-UX 版,HI-UX/WE2 版)及び Intel fortran コンパイラ for 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。
- Fortran 直接編成ファイルのレコード長の指定は、１バイト単位で指定しています。処理系によっては指定する単位が異なる場合が有りますのでご注意ください。

（２）次のコマンドを入力することにより、３か月予報メンバー別格子点値、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る。

実行例）$ dcd_3mepac

- デコードするファイル名は、サンプルプログラムの34行目に記述されています。
  filename='3MEGPV'
- デコードされたデータは、要素ごと、予報時間ごとに、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。ファイル名には48行目で読み込むデータ名称dnamen を使用します。
- ファイルの出力を止めたい場合は、サンプルプログラム33行目の
  ifout=1
  を
  ifout=0
  など、1以外の数値に変更してください。

※Linux は、Linus Torvalds の米国及びその他の国における登録商標あるいは商標です。
※HP-UX は、米国 Hewlett-Packard Company のオペレーティングシステムの名称です。
※HI-UX/WE2 は、(株)日立製作所の登録商標です。
※RedHat は、米国 Red Hat Software,Inc.の登録商標です。

別紙

解読サンプルプログラムのソースコード
（３か月予報メンバー別格子点値ファイル用）

```
c=======================================================================
c
c   Sample Code to read Seasonal ensemble forcast GPV of JMA
c                                        2003.04.18 JMA/CPD
c
c       Tested on Intel fortran compiler for Linux(little endian)
c                and HITACHI fotran compiler for HI-UX(big endian)
c=======================================================================
        program dcd_3me_mem

        parameter(mxlen=10000000)
        parameter(mtg=10,mtout=20)

        character(1)          :: bul(mxlen)
        character(4)          :: cgrib
        integer(4)            :: iend,len,ngrib
        character(4)          :: rname,yobi
        character(80)         :: sakusei
        character(16)         :: yobi1,syubetu
        character(12)         :: cdate,syurui
        character(124)        :: yobi2
        character(8)          :: yobi3
        character(20)         :: dnamen
        character(12)         :: syurui
        integer(4)            :: lrecn
        character(30)         :: filename
        integer(4)            :: ifout

        call checkendian(iend)

        len=0;ngrib=0

        ifout=1                     !  1:Output data file by 4bytes real
        filename='3MEGPV'           !  Input file name
        open(mtg,file=filename,access='sequential',form='unformatted')

        do
        read(mtg,end=99)rname,lrecn
        backspace mtg
        if(rname == 'VREC')then
          read(mtg)rname,lrec,yobi,sakusei,iver,yobi1
          write(*,*)rname,lrec,yobi,sakusei,iver,yobi1
          write(*,*)'START RECORD !!'
        else if(rname == 'CNTL')then
          read(mtg)rname,lrec,yobi,syubetu,cdate,initime,yobi2
          write(*,*)rname,lrec,yobi,syubetu,cdate,initime,yobi2
        else if(rname == 'DATA')then
          read(mtg)rname,lrecn,yobi,dnamen,syurui,bul(1:lrecn-44)
          write(*,*)rname,lrecn,yobi,dnamen,syurui
          icount=1
          ipos=0
```

```
          do
            call cpbytes(bul(ipos+1),    1,    4, cgrib,      1,      1)
            if(cgrib.ne.'GRIB') exit
            call cpbytes(bul(ipos+1),    5,    3,      len,      2, iend)
            call cpbytes(bul(ipos+1),    8,    1, ngrib,      4, iend)
            call dcdgrib1(bul(ipos+1),len,icount, iend,dnamen,ifout)
            ipos=ipos+len
          end do
        else if(rname == 'END')then
          read(mtg)rname,lrec,yobi,yobi3
          write(*,*)rname,lrec,yobi,yobi3
          write(*,*)'END RECORD !!'
        else
          write(*,*)'unsupported record!! rname=',rname
          stop 80
        endif

        end do

        close(mtg)

        stop
 99     write(*,*)'FILE READ END!!'
        stop 10
        end
c-----------------------------------------------------------------
c     Decode GRIB1
c-----------------------------------------------------------------
        subroutine dcdgrib1(bul,len,icount,iend,dnamen,ifout)

c         parameter(imax=144,jmax=73,ijmax=imax*jmax,lmap=ijmax/8+1)
        parameter(imax=144,jmax=37,ijmax=imax*jmax,lmap=ijmax/8)
        character(1)     :: bul(len),bitmap(lmap)
        character(4)     :: end
        character(20)    :: dnamen
        real(4)          :: gpv(ijmax)
        integer(4)       :: ibit(imax,jmax)
        integer(4)       :: ifg, ifb,iprm,ilvl,lvel,kds,kde,iend
        integer(4)       :: ifout

        call dcdsect0(bul(     1)        ,     8, iend)
        call dcdsect1(bul( 8+1)        ,    28,     ifg, ifb,
       &                                    iprm, ilvl, lvel,    kds,    kde, iend)
        if(ifg.eq.1)then
            call dcdsect2(bul(36+1),    32,    imxo,jmxo,la1, iend)
        endif
        if(ifb.eq.1)then
            call dcdsect3(bul(68+1),len-68,len3,ibit,lodd,imxo,jmxo, iend)
        else
            len3=0
        endif
          call dcdsect4(bul(68+len3+1),len4,ibit,gpv,imxo,jmxo,
       &                          ifg,ifb,iend)
```

```
           call dcdsect5(bul(len-4+1)       , 4)
c---------------
c     Check gpv
c          call filn(fil,ifg,la1,iprm,ilvl,lvel,kds,kde)
          write(*,*)
          write(*,*)'DATA Check !!'
          ii=52
          do 100 j=1,jmxo
              write(*,*)dnamen,'gpv(',ii,',',j,')=',gpv(imxo*(j-1)+ii)
   100 continue
c---------------
c     Output to file
           if( ifout == 1)then
              mtgr=30
              open(mtgr,file=dnamen,access='direct',form='unformatted',
       &                                                    recl=imxo*jmxo*4)
              write(mtgr,rec=1)gpv
              close(mtgr)
           endif
c---------------

          icount=icount+1
          return
          end
c-------------------------------------------------------------
c    THE INDICATOR SECTION
c-------------------------------------------------------------
          subroutine dcdsect0(bul,len0, iend)

          character(1)    :: bul(len0)
          character(4)    :: cgrib
          integer(4)      :: len0,len,ngrib,iend

          len=0;ngrib=0

          call cpbytes(bul,    1,    4, cgrib,     1,       1) ! 'GRIB'
          call cpbytes(bul,    5,    3,     len,     2, iend) ! Total length,
                                                              !                 in octets
          call cpbytes(bul,    8,    1, ngrib,     4, iend) ! Edition number
                                                              !           - currently 1

          write(*,*)'Print Section 0'
          write(*,*)cgrib
          write(*,*)len,': Total length, in octets'
          write(*,*)ngrib,': Edition number'

          return
          end
c-------------------------------------------------------------
c     THE PRODUCT DEFINITION SECTION
c-------------------------------------------------------------
          subroutine dcdsect1(bul,len1,ifg,ifb,iprm,ilvl,lvel,kds,kde,iend)
```

```
  character(1)   :: bul(len1)
  character(20) :: fil
  integer(4)      :: iff,iflg,ifg,ifb
  integer(4)      :: len,ng,jma,np,iprm,ilvl,lvel,iyy,imon,idy,ihr
&           ,imin,iunit,kds,kde,idx,nnn,nn2,icn,jma2,idec,iend

  len=0;ng=0;jma=0;np=0;iff=0;iflg=0;iprm=0;ilvl=0;lvel=0
  iyy=0;imon=0;idy=0;ihr=0;imin=0;iunit=0;kds=0;kde=0
  idx=0;nnn=0;nn2=0;icn=0;jma2=0;idec=0;ifb=0;ifg=0

  call cpbytes(bul,   1,   3,   len,   2, iend)
  call cpbytes(bul,   4,   1,    ng,   4, iend)
  call cpbytes(bul,   5,   1,   jma,   4, iend)
  call cpbytes(bul,   6,   1,    np,   4, iend)
  call cpbytes(bul,   7,   1,   iff,   4, iend)
  call cpbytes(bul,   8,   1, iflg,   4, iend)
  call cpbytes(bul,   9,   1, iprm,   4, iend)
  call cpbytes(bul, 10,   1, ilvl,   4, iend)
  call cpbytes(bul, 11,   2, lvel,   3, iend)
  call cpbytes(bul, 13,   1,   iyy,   4, iend)
  call cpbytes(bul, 14,   1, imon,   4, iend)
  call cpbytes(bul, 15,   1,   idy,   4, iend)
  call cpbytes(bul, 16,   1,   ihr,   4, iend)
  call cpbytes(bul, 17,   1, imin,   4, iend)
  call cpbytes(bul, 18,   1,iunit,   4, iend)
  call cpbytes(bul, 19,   1,   kds,   4, iend)
  call cpbytes(bul, 20,   1,   kde,   4, iend)
  call cpbytes(bul, 21,   1,   idx,   4, iend)
  call cpbytes(bul, 22,   2,   nnn,   3, iend)
  call cpbytes(bul, 24,   1,   nn2,   4, iend)
  call cpbytes(bul, 25,   1,   icn,   4, iend)
  call cpbytes(bul, 26,   1, jma2,   4, iend)
  call cpbytes(bul, 27,   2, idec,   3, iend)

     call mvbits(iflg,   6,   1,   ifb,   0)
     call mvbits(iflg,   7,   1,   ifg,   0)

  write(*,*)'Print Section 1'

  write(*,*)len   ,': Length in octets'
  write(*,*)ng    ,': Parameter Table Version number'
  write(*,*)jma   ,': Identification of center(34:jma)'
  write(*,*)np    ,': Generating process ID number'
  write(*,*)iff   ,': Grid Identification '
  write(*,*)iflg ,': Flag'
  write(*,*)iprm ,': Indicator of parameter and units '
  write(*,*)ilvl ,': Indicator of type of level or layer'
  write(*,*)lvel ,': Height, pressure, etc'
  write(*,*)iyy   ,': Year of century'
  write(*,*)imon ,': Month of year'
  write(*,*)idy   ,': Day of month '
  write(*,*)ihr   ,': Hour of day'
  write(*,*)imin ,': Minute of hour'
```

```
      write(*,*)iunit,': Forecast time unit'
      write(*,*)kds   ,': P1 - Period of time'
      write(*,*)kde   ,': P2 - Period of time'
      write(*,*)idx   ,': Time range indicator'
      write(*,*)nnn   ,': Number included in average'
      write(*,*)nn2   ,': Number Missing from averages'
      write(*,*)icn   ,': Century of Initial time'
      write(*,*)jma2 ,': Identification of sub-center '
      write(*,*)idec ,': The decimal scale factor D'

      write(*,*)ifg ,': GDS 0:Omitted 1:Included'
      write(*,*)ifb ,': BMS 0:Omitted 1:Included'

      return
      end
c-----------------------------------------------------------------------
c     GRID DESCRIPTION SECTION
c-----------------------------------------------------------------------
      subroutine dcdsect2(bul,len2,imx,jmx,la1, iend)

      character(1)    :: bul(len2)
      integer(4)      :: len2
      integer(4)      :: len,nv,ipv,ityp,imx,jmx,la1,lo1,ifr,la2,lo2
     &                        ,id,jd,iflg,iend

      len=0;nv=0;ipv=0;ityp=0;imx=0;jmx=0;la1=0;lo1=0;ifr=0
      la2=0;lo2=0;id=0;jd=0;iflg=0

      call cpbytes(bul,   1,   3,   len,   2, iend)
      call cpbytes(bul,   4,   1,    nv,   4, iend)
      call cpbytes(bul,   5,   1,   ipv,   4, iend)
      call cpbytes(bul,   6,   1, ityp,    4, iend)
c   Grid Definitions
      call cpbytes(bul,   7,   2,   imx,   3, iend)
      call cpbytes(bul,   9,   2,   jmx,   3, iend)
      call cpbytes(bul, 11,   3,   la1,   2, iend)
      call cpbytes(bul, 14,   3,   lo1,   2, iend)
      call cpbytes(bul, 17,   1,   ifr,   4, iend)
      call cpbytes(bul, 18,   3,   la2,   2, iend)
      call cpbytes(bul, 21,   3,   lo2,   2, iend)
      call cpbytes(bul, 24,   2,    id,   3, iend)
      call cpbytes(bul, 26,   2,    jd,   3, iend)
      call cpbytes(bul, 28,   1, iflg,    4, iend)

      write(*,*) 'Section 2'

      write(*,*)len,': Length in octets of the Grid Description Section'
      write(*,*)nv ,': the number of vertical coordinate parameters'
      write(*,*)ipv,': the location of the list of vertical coordinate '
      write(*,*)ityp ,': Data representation type '
      write(*,*)imx   ,': No. of points along a latitude circle'
      write(*,*)jmx   ,': No. of points along a longitude meridian'
      write(*,*)la1   ,': latitude of first grid point units'
```

```
      write(*,*)lo1    ,': longitude of first grid point units'
      write(*,*)ifr    ,': Resolution and component flags'
      write(*,*)la2    ,': Latitude of last grid point'
      write(*,*)lo2    ,': Longitude of last grid point'
      write(*,*)id     ,': Longitudinal Direction Increment'
      write(*,*)jd     ,': Latitudinal Direction Increment'
      write(*,*)iflg ,': Scanning mode flags'

      return
      end
c-----------------------------------------------------------------------
c       BIT MAP SECTION
c-----------------------------------------------------------------------
      subroutine dcdsect3(bul,len,len3,ibit,lodd,imxo,jmxo,iend)

      character(1)     :: bul(len)
      integer(4)       :: ibit(imxo*jmxo)
      integer(4)       :: len3,lodd,imxo,jmxo,iend

      len3=0;lodd=0;nu=0

      call cpbytes(bul,    1,    3, len3,    2, iend) ! Length in octets of
                                                      !     Bit Map Section
      call cpbytes(bul,    4,    1, lodd,    4, iend) ! Number of unused bits
      call cpbytes(bul,    5,    2,      nu,    3, iend) ! = 0: a bit map follows

      write(*,*)'Section 3'
      write(*,*)len3,'Length in octets of Bit Map Section'
      write(*,*)lodd,' Number of unused bits'
      write(*,*)nu,'= 0: a bit map follows'

      non=0
      noff=0

      nx=imxo*jmxo
      lnb=(len3-6)/4+1
      do 100 n=1,nx
         if(mod(n,8).eq.0)then
            nb=n/8
            nodd=8
         else
            nb=n/8+1
            nodd=mod(n,8)
         endif
         call cpbytes(bul, 6+nb,    4, iwk,    1, iend)
         call mvbits(iwk, 32-nodd,    1,ibit(n),    0)
         if(ibit(n).eq.0) then
            noff=noff+1
         else
            non=non+1
         endif
  100 continue
```

```
      return
      end
c-------------------------------------------------------------
c     BINARY DATA SECTION
c-------------------------------------------------------------
      subroutine dcdsect4(bul,len,ibit,gpv,
     &                    imxo,jmxo,ifg,ifb,iend)

      integer(4)   :: len,ifgodd,iflg,lodd,imxo,jmxo,ifg,ifb
      integer(4)   :: ibit(imxo*jmxo)
      integer(4)   :: ibitl,igpv,iend
      integer(2)   :: jexp,jexp2
      character(1):: bul(10000000)
      character(4):: cgpvmin
      real(4)      :: gpv(imxo*jmxo),gpvmin,fmiss

      fmiss=-99999
      len=0;ifgodd=0;ibitl=0
      iflg=0;lodd=0
      jexp2=0;jexp=0

      call cpbytes(bul,   1,   3,    len,   2, iend)
      call cpbytes(bul,   4,   1,ifgodd,    4, iend)

        call mvbits(ifgodd,4,   4,   iflg,   0)
        call mvbits(ifgodd,0,   4,   lodd,   0)
      if(iend.eq.1)then
        call cpbytes(bul,   5,   2,   jexp,   1,    1)
      else
        call cpbytes(bul,   5,   1,   jexp,   2,    1)
        call cpbytes(bul,   6,   1,   jexp,   1,    1)
      endif

      if(jexp.lt.0)then
        jexp2=jexp*2
        jexp2=jexp2/2
        jexp=jexp2*(-1)
      endif
      call cpbytes(bul,   7,   4,cgpvmin, 1,      1)
      call cpbytes(bul, 11,   1,   ibitl, 4, iend)
      call vos2ieee(cgpvmin,gpvmin,iend)
      if(ifb.eq.0) ibit=1
      ipos=1
      do 100 n=1,imxo*jmxo
        if(ibit(n).eq.1)then
          call cpbits(bul(12),   ipos, ibitl, igpv,   0, iend)
          gpv(n)=igpv*(2.0**jexp)+gpvmin
          ipos=ipos+ibitl
        else
          gpv(n)=fmiss
        endif
  100 continue
```

```
        return
        end
c-------------------------------------------------------------
c   END SECTION
c-------------------------------------------------------------
        subroutine dcdsect5(bul, len)

        character(len)   :: bul
        integer(4)       :: len

        write(*,*)'Print Sectsion 5   ',bul

        return
        end
c=============================================================
c
c     TOOLS
c
c-------------------------------------------------------------
c       copy len bytes from cin -> cout
c
        subroutine cpbytes(cin,from,len,cout,to,iend)

        integer(4)     :: from,len,to,iend,tobi
        character(1) :: cin(from+len-1),cout(to+len-1)

        if(iend.ne.1.and.iend.ne.-1)then
            write(*,*)'Parameter iend is incorrect'
            write(*,*)'iend must be 1(Big endian) or -1(Little endian)'
            stop 123
        endif

        if(iend.eq.-1)then
            tobi=5-to
        else
            tobi=to
        endif

        do 100 l=1,len
            write(cout(tobi+iend*(l-1)),'(a1)')cin(from+l-1)
   100 continue

        return
        end
c-------------------------------------------------------------
c     Convert M -> ieee   (real(4))
c           iend=1 : BIG ENDIAN
c           iend=-1: LITTLE ENDIAN
c
        subroutine vos2ieee(ivos,real,iend)

        real(4)          :: real
        integer(4)     :: ia,ib,is,ivos
```

```
      ia=0;ib=0;is=0

      if(iend.eq.1)then
          call mvbits(ivos,   24,    7,   ia, 0)
          call mvbits(ivos,    0,   24,   ib, 0)
          call mvbits(ivos,   31,    1,   is, 0)
      else
          call mvbits(ivos,    0,    7,   ia,   0)
          call mvbits(ivos,   24,    8,   ib,   0)
          call mvbits(ivos,   16,    8,   ib,   8)
          call mvbits(ivos,    8,    8,   ib, 16)
          call mvbits(ivos,    7,    1,   is,   0)
      endif
      real=2.**(-24)*ib*16**(ia-64)
      if(is.eq.1) real=(-1)*real

      return
      end
c-----------------------------------------------------------------
c     copy ibitl bits from bul -> igpv
c
      subroutine cpbits(bul,ipos,ibitl,igpv,ito,iend)

      character(1)   :: bul((ipos+ibitl)/8+4)
      integer(4)     :: ipos,ibitl,igpv,ito,iend
      integer(4)     :: lposi,iwk


      igpv=0
      if(mod(ipos,8).eq.0)then
          nbyt=ipos/8
          lodd=8
      else
          nbyt=ipos/8+1
          lodd=mod(ipos,8)
      endif


          call cpbytes(bul,nbyt,   4,iwk,   1, iend)
      do 10 i=1,ibitl
          lposi=32-(lodd+ibitl-1)+i-1
          call mvbits(   iwk, lposi,   1, igpv,   i-1)
   10   continue

      return
      end
c-----------------------------------------------------------
c   Identify element
c
      subroutine filn(fil,ifg,la1,iprm,ilvl,lvel,kds,kde)

      character(20)   :: fil
```

```
integer(4)        :: ifg,la1,iprm,ilvl,lvel,kds,kde

if        (iprm.eq.007.and.ilvl.eq.100.and.lvel.eq.500)then
    write(fil,'(a9)') 'Z500____x'
else if(iprm.eq.027.and.ilvl.eq.100.and.lvel.eq.500)then
    write(fil,'(a9)') 'Z500ANOMx'
else if(iprm.eq.142.and.ilvl.eq.100.and.lvel.eq.500)then
    write(fil,'(a9)') 'Z500SPRDx'
else if(iprm.eq.140.and.ilvl.eq.100.and.lvel.eq.500)then
    write(fil,'(a9)') 'Z500HANMx'
else if(iprm.eq.011.and.ilvl.eq.100.and.lvel.eq.850)then
    write(fil,'(a9)') 'T850____x'
else if(iprm.eq.025.and.ilvl.eq.100.and.lvel.eq.850)then
    write(fil,'(a9)') 'T850ANOMx'
else if(iprm.eq.143.and.ilvl.eq.100.and.lvel.eq.850) then
    write(fil,'(a9)') 'T850SPRDx'
else if(iprm.eq.002.and.ilvl.eq.102.and.lvel.eq.000) then
    write(fil,'(a9)') 'PSEA____x'
else if(iprm.eq.141.and.ilvl.eq.102.and.lvel.eq.000) then
    write(fil,'(a9)') 'PSEASPRDx'
else if(iprm.eq.026.and.ilvl.eq.102.and.lvel.eq.000) then
    write(fil,'(a9)') 'PSEAANOMx'
else if(iprm.eq.061.and.ilvl.eq.001.and.lvel.eq.000) then
    write(fil,'(a9)') 'RAIN____x'
else if(iprm.eq.052.and.ilvl.eq.100.and.lvel.eq.850) then
    write(fil,'(a9)') 'H850____x'
else if(iprm.eq.033.and.ilvl.eq.100.and.lvel.eq.850) then
    write(fil,'(a9)') 'U850____x'
else if(iprm.eq.034.and.ilvl.eq.100.and.lvel.eq.850) then
    write(fil,'(a9)') 'V850____x'
else if(iprm.eq.033.and.ilvl.eq.100.and.lvel.eq.200) then
    write(fil,'(a9)') 'U200____x'
else if(iprm.eq.034.and.ilvl.eq.100.and.lvel.eq.200) then
    write(fil,'(a9)') 'V200____x'
else if(iprm.eq.007.and.ilvl.eq.100.and.lvel.eq.100) then
    write(fil,'(a9)') 'Z100____x'
else if(iprm.eq.027.and.ilvl.eq.100.and.lvel.eq.100) then
    write(fil,'(a9)') 'Z100ANOMx'
else
    write(6,*)'unit cannot be defined!!'
    write(*,*)'iprm,ilvl,lvel',iprm,ilvl,lvel
    stop 888
endif

if(ifg.eq.1)then
    if(la1.gt.0)then
        write(fil(9:9),'(a1)')'N'
    else
        write(fil(9:9),'(a1)')'S'
    endif
endif

if(kds.lt.10) then
```

```
            write(fil(10:11),'(a1,i1)')'0',kds
        else
            write(fil(10:11),'(i2)')kds
        endif

        if(kde.lt.10)then
            write(fil(12:13),'(a1,i1)')'0',kde
        else
            write(fil(12:13),'(i2)')kde
        endif

        write(fil(14:17),'(a4)')'.dat'

        return
        end
c-----------------------------------------------------------------
c       Identify endian
c
c          iend   1 : BIG ENDIAN
c                -1 : LITTLE ENDIAN
        subroutine checkendian(iend)

        integer(4)   :: iend,ia,ib,il

        ia=1
        ib=0;il=0

        call cpbytes(ia,4,1,ib,4,1)
        call cpbytes(ia,1,1,il,1,1)

        if(ib.eq.1)then
            iend=1
        else if(il.eq.1)then
            iend=-1
        else
            write(*,*)'UNKNOWN ENDIAN ??'
            stop 345
        end if

        return
        end
```