

CINRAD Support in the ARPS Remapping Program – 88d2arps

1. Introduction

There are two ways to decode Chinese netrad (CINRAD) radar data with the remapping program – 88d2arps. One is the compile-time modification which was provided by Chunxi Zhang from the Peking University in China. Another is a run-time option which was provided by an anonymous user (“z kf”).

2. Usages

Before running the program 88d2arps, user is required to check and modify the radar information file `data/adas/radarinfo_CINRAD.dat`. Then he/she can set the environment variable for 88d2arps as

csh/tcsh shell:

```
setenv RADARFILE $ARPSROOT/data/adas/radarinfo_CINRAD.dat
```

bash shell

```
export RADARFILE=$ARPSROOT/data/adas/radarinfo_CINRAD.dat
```

where `$ARPSROOT` is the root directory of the ARPS package.

2.1 Compile-time modification

Users should find the following line

```
/* #define      ALLOW_PKU */
```

in file `src/88d2arps/a2io/config.h` and uncomment it by removing the leading “/*” and the ending “*/”.

Then the program 88d2arps can be compiled and run with CINRAD data just as it is instructed in file `docs/88d2arps50.pdf`.

2.2 Run-time option

Users compile program 88d2arps just as it is instructed in file `docs/88d2arps50.pdf`.

When running the program, users should provide an option “-rad98”. For example,

```
../bin/88d2arps KTAN -rad98 -diskf radarfile_china < adas.input
```

Appendices

The following appendices about CINRAD data formats and code for decoding (program 88d2arps does not use it, it is provided just for reference) were provided by “z kf”.

— Monday, July 30, 2007

附录 1: CINRAD SA/SB 雷达基数据格式

字节顺序	双字节顺序	数据类型	说明	
1-14	1-7		保留	雷达信息头 (28 字节)
15-16	8	2 字节	1-表示雷达数据	
17-28	9-14		保留	
29-32	15-16	4 字节	径向数据收集时间(毫秒,自 00:00 开始)	
33-34	17	2 字节	儒略日 (Julian) 表示, 自 1970 年 1 月 1 日开始	
35-36	18	2 字节	不模糊距离 (表示: 数值/10.=千米)	
37-38	19	2 字节	方位角 (编码方式: [数值/8.]*[180./4096.]=度)	
39-40	20	2 字节	当前仰角内径向数据序号	
41-42	21	2 字节	径向数据状态 0: 该仰角的第一条径向数据 1: 该仰角中间的径向数据 2: 该仰角的最后一条径向数据 3: 体扫开始的第一条径向数据 4: 体扫结束的最后一条径向数据	
43-44	22	2 字节	仰角 (编码方式: [数值/8.]*[180./4096.]=度)	
45-46	23	2 字节	体扫内的仰角数	
47-48	24	2 字节	反射率数据的第一个距离库的实际距离(单位:米)	
49-50	25	2 字节	多普勒数据的第一个距离库的实际距离(单位:米)	
51-52	26	2 字节	反射率数据的距离库长 (单位: 米)	
53-54	27	2 字节	多普勒数据的距离库长 (单位: 米)	
55-56	28	2 字节	反射率的距离库数	
57-58	29	2 字节	多普勒的距离库数	
59-60	30	2 字节	扇区号	
61-64	31-32	4 字节	系统订正常数	
65-66	33	2 字节	反射率数据指针 (偏离雷达数据信息头的字节数) 表示第一个反射率数据的位置	
67-68	34	2 字节	速度数据指针 (偏离雷达数据信息头的字节数) 表示第一个速度数据的位置	
69-70	35	2 字节	谱宽数据指针 (偏离雷达数据信息头的字节数) 表示第一个谱宽数据的位置	
71-72	36	2 字节	多普勒速度分辨率。 2: 表示 0.5 米/秒 4: 表示 1.0 米/秒	
73-74	37	2 字节	体扫 (VCP) 模式 11: 降水模式, 16 层仰角 21: 降水模式, 14 层仰角 31: 晴空模式, 8 层仰角 32: 晴空模式, 7 层仰角	
75-82	38-41		保留	
83-84	42	2 字节	用于回放的反射率数据指针, 同 33	
85-86	43	2 字节	用于回放的速度数据指针, 同 34	
87-88	44	2 字节	用于回放的谱宽数据指针, 同 35	

89-90	45	2 字节	Nyquist 速度（表示：数值/100. = 米/秒）	
91-128	46-64		保留	
129-588	65-294	1 字节	反射率 距离库数：0-460 编码方式：（数值-2）/2.-32 = DBZ 当数值为 0 时，表示无回波数据（低于信噪比阈值） 当数值为 1 时，表示距离模糊	基数据 部分 (2300 字节)
129-1508	65-754	1 字节	速度 距离库数：0-920 编码方式： 分辨率为 0.5 米/秒时 （数值-2）/2.-63.5 = 米/秒 分辨率为 1.0 米/秒时 （数值-2）-127 = 米/秒 当数值为 0 或 1 时，意义同上	
129-2428	65-1214	1 字节	谱宽 距离库数：0-920 编码方式： （数值-2）/2.-63.5 = 米/秒 当数值为 0 或 1 时，意义同上	
2429-2432	1215-1216		保留	

说明：

1. 数据的存储方式
每个体扫存储为一个单独的文件
2. 数据的排列方式
按照径向数据的方式顺序排列，对于 CINRAD SA/SB 雷达，体扫数据排列自低仰角开始到高仰角结束。
3. 径向数据的长度
径向数据的长度固定，为 2432 字节。
4. 距离库长和库数
反射率距离库长为 1000 米，最大距离库数为 460；
速度和谱宽距离库长为 250 米，最大距离库数为 920。

附录 2: CINRAD CB 雷达基数据格式

字节顺序	双字节顺序	数据类型	说明
1-14	1-7		保留
15-16	8	2 字节	1-表示雷达数据
17-28	9-14		保留
29-32	15-16	4 字节	径向数据收集时间(毫秒,自 00:00 开始)
33-34	17	2 字节	儒略日 (Julian) 表示, 自 1970 年 1 月 1 日开始
35-36	18	2 字节	不模糊距离 (表示: 数值/10.=千米)
37-38	19	2 字节	方位角 (编码方式: [数值/8.]*[180./4096.] =度)
39-40	20	2 字节	当前仰角内径向数据序号
41-42	21	2 字节	径向数据状态 0: 该仰角的第一条径向数据 1: 该仰角中间的径向数据 2: 该仰角的最后一条径向数据 3: 体扫开始的第一条径向数据 4: 体扫结束的最后一条径向数据
43-44	22	2 字节	仰角 (编码方式: [数值/8.]*[180./4096.] =度)
45-46	23	2 字节	体扫内的仰角数
47-48	24	2 字节	反射率数据的第一个距离库的实际距离(单位:米)
49-50	25	2 字节	多普勒数据的第一个距离库的实际距离(单位:米)
51-52	26	2 字节	反射率数据的距离库长 (单位: 米)
53-54	27	2 字节	多普勒数据的距离库长 (单位: 米)
55-56	28	2 字节	反射率的距离库数
57-58	29	2 字节	多普勒的距离库数
59-60	30	2 字节	扇区号
61-64	31-32	4 字节	系统订正常数
65-66	33	2 字节	反射率数据指针 (偏离雷达数据信息头的字节数) 表示第一个反射率数据的位置
67-68	34	2 字节	速度数据指针 (偏离雷达数据信息头的字节数) 表示第一个速度数据的位置
69-70	35	2 字节	谱宽数据指针 (偏离雷达数据信息头的字节数) 表示第一个谱宽数据的位置
71-72	36	2 字节	多普勒速度分辨率。 2: 表示 0.5 米/秒 4: 表示 1.0 米/秒
73-74	37	2 字节	体扫 (VCP) 模式 11: 降水模式, 16 层仰角 21: 降水模式, 14 层仰角 31: 晴空模式, 8 层仰角 32: 晴空模式, 7 层仰角
75-82	38-41		保留
83-84	42	2 字节	用于回放的反射率数据指针, 同 33
85-86	43	2 字节	用于回放的速度数据指针, 同 34
87-88	44	2 字节	用于回放的谱宽数据指针, 同 35
89-90	45	2 字节	Nyquist 速度 (表示: 数值/100. = 米/秒)

91-128	46-64		保留	
129-928	65-464	1 字节	反射率 距离库数: 0-800 编码方式: (数值-2) / 2. -32 = DBZ 当数值为 0 时, 表示无回波数据 (低于信噪比阈值) 当数值为 1 时, 表示距离模糊	基数据 部分 (4000 字节)
129-2528	65-1264	1 字节	速度 距离库数: 0-1600 编码方式: 分辨率为 0.5 米/秒时 (数值-2) / 2. -63.5 = 米/秒 分辨率为 1.0 米/秒时 (数值-2) - 127 = 米/秒 当数值为 0 或 1 时, 意义同上	
129-4128	65-2064	1 字节	谱宽 距离库数: 0-1600 编码方式: (数值-2) / 2. -63.5 = 米/秒 当数值为 0 或 1 时, 意义同上	
4129-4132	1215-2066		保留	

说明:

5. 数据的存储方式

每个体扫存储为一个单独的文件

6. 数据的排列方式

按照径向数据的方式顺序排列, 对于 CINRAD CB 雷达, 体扫数据排列自低仰角开始到高仰角结束。

7. 径向数据的长度

径向数据的长度固定, 为 4132 字节。

8. 距离库长和库数

反射率距离库长为 500 米, 最大距离库数为 800;

速度和谱宽距离库长为 125 米, 最大距离库数为 1600。

附录 3: 程序中的重要数据说明

1. 文件名

Filename[], 输入需要读取的基数据的文件名。需将该文件放在执行程序所在的目录中才能读出其中的数据。

2. 保存反射率、速度、谱宽, 各层仰角的数组。文件中读取的基数据存放在下列数组中:

```
float VolRef[MaxCuts][MaxRads][RGates];           //反射率(浮点型, 单位: DBZ)
float VolVel[MaxCuts][MaxRads][VGates];           //速度(浮点型, 单位: M/S)
float VolSpw[MaxCuts][MaxRads][WGates];           //谱宽(浮点型, 单位: M/S)
float Elvation[MaxCuts];                           //各层仰角(浮点型, 单位: 度)
数组中无效数据标记为-999.0, 距离折叠标记为 999.0。
```

其中,

- 1) MaxCuts=20, 为最大层数;
- 2) MaxRads 为方位数, 每度保存一个径向;
- 3) Rgates 为每个径向上反射率的距离库数, C 波段为 800, 对应分辨率为 0.5 公里; S 波段为 460, 对应分辨率为 1 公里;
- 4) Vgates 为每个径向上径向速度的距离库数, C 波段为 1600, 对应分辨率为 0.125 公里; S 波段为 920, 对应分辨率为 0.25 公里;
- 5) Wgates 为每个径向上谱宽的距离库数, C 波段为 1600, 对应分辨率为 0.125 公里; S 波段为 920, 对应分辨率为 0.25 公里;

3. 读取不同波段的基数据文件的方法

在头文件 DataFormat.h 中, 对距离库数的定义为, 用来读取 **S** 波段的基数据:

```
const int RGates = 460;      //反射率距离库数
const int VGates = 920;      //速度距离库数
const int WGates = 920;      //谱宽距离库数
```

若要读取 **C** 波段的基数据时, 只需将上述定义修改为:

```
const int RGates = 800;      //反射率距离库数
const int VGates = 1600;     //速度距离库数
const int WGates = 1600;     //谱宽距离库数
```

注意:

1) 关于仰角层的说明:

SA, SB, CB 雷达在低层每个仰角上扫描两次, 程序中, 在保存基数据到数组中时, 记为一个仰角层。以 21 扫描模式为例, VCP 仰角为:

0.5, 0.5, 1.5, 1.5, 2.4, 3.4, 4.3, 6.0, 9.9, 14.6, 19.5 ----11 个 PPI 扫描

其中 0.5 和 1.5 分别扫描 2 次, 记为一个仰角, 因此, 数组 Elvation[] 中有 9 个有效元素, 为:

0.5, 1.5, 2.4, 3.4, 4.3, 6.0, 9.9, 14.6, 19.5

相应的, 基数据 9 层有效。

2) 数组中无效数据记为-999.0, 距离折叠标记为 999.0。

```

//*****//
/* 名称: ReadBaseData(char filename[80])
/* 类型: 布耳型, 若读文件操作失败 (体扫不完整等), 返回 FALSE。
/* 功能: 从基数据文件中读出仰角, 反射率, 速度, 谱宽数据, 保存到
/*      数组中。
/* 参数: filename, 基数据文件名
//*****//
bool ReadBaseData(char filename[80])
{
FILE *fp=0;

int FstBin, LstBin, BinNum;
float CurAz, CurEl;
int ElIndex, AzIndex, BnIndex;
int ptrPos;

size_t readSize;
int fileEndFlag;

bool VolBeg=false;
bool VolEnd=false;

bool RFlag, VFlag, WFlag;

fp = fopen(filename, "rb");
if(fp==0) return false;

pOneRadial = 0;
pOneRadial = (RADIALDATA*)malloc(sizeof(RADIALDATA));
if(pOneRadial==0) return false;

//Initialize array
for(ElIndex=0; ElIndex<MaxCuts; ElIndex++)
{
for(AzIndex=0; AzIndex<MaxRads; AzIndex++)
{
for(BnIndex=0; BnIndex<RGates; BnIndex++)
RDData[ElIndex][AzIndex][BnIndex] = VALUE_INVALID;
for(BnIndex=0; BnIndex<VGates; BnIndex++)
VData[ElIndex][AzIndex][BnIndex] = VALUE_INVALID;
for(BnIndex=0; BnIndex<WGates; BnIndex++)
WData[ElIndex][AzIndex][BnIndex] = VALUE_INVALID;
} //end az
}

```

```

    Elevation[ElIndex]=VALUE_INVALID;
} //end el

do
{
    //Initialize flags
    RFlag=VFlag=WFlag=false;

    readSize = fread(pOneRadial, sizeof(RADIALDATA), 1, fp);
    fileEndFlag = feof(fp);

    //Start a volume scan
    if(pOneRadial->RadialStatus == VOL_BEG)
    {
        ElIndex=0;
        CurEl = float((pOneRadial->El/8.)*(180./4096.));
        Elevation[ElIndex] = CurEl;

        VolBeg = true;
        //output text information
        // TRACE("VCP number is %3d\\n", pOneRadial->VcpNumber);
        printf("VCP number is %3d\\n", pOneRadial->VcpNumber);

        // TRACE("Elevation %3d (%5.2f Degree) start...\\n", ElIndex+1, CurEl);
        printf("Elevation %3d (%5.2f Degree) start...\\n", ElIndex+1, CurEl);
    }

    //Find the beginning of the volume scan
    if(!VolBeg) continue;

    //Start an elevation
    if(pOneRadial->RadialStatus == ELV_BEG)
    {
        CurEl = float((pOneRadial->El/8.)*(180./4096.));
        if(CurEl-Elevation[ElIndex] > 0.4)
        { //different elevation angle
            ElIndex++;
            Elevation[ElIndex] = CurEl;
        }
        // TRACE("Elevation %3d (%5.2f Degree) start...\\n", ElIndex+1, CurEl);
        printf("Elevation %3d (%5.2f Degree) start...\\n", ElIndex+1, CurEl);
    }

    //Start an elevation
    if(pOneRadial->RadialStatus == VOL_END)

```



```

VolEnd=true;

// if(pOneRadial->RadialStatus == ELV_BEG || pOneRadial->RadialStatus == VOL_BEG)

//Calculate azimuth angle and Azimuth Index
CurAz = float((pOneRadial->Az/8.)*(180./4096.));
if(CurAz >= 360.) CurAz = CurAz-360.;
AzIndex = int(CurAz+0.5);

//what kind of data in this cut
if(pOneRadial->PtrOfReflectivity !=0) RFlag=true;
if(pOneRadial->PtrOfVelocity !=0) VFlag=true;
if(pOneRadial->PtrOfSpectrumWidth !=0) WFlag=true;

//Save reflectivity data into the array
if(RFlag)
{
    //Get first bin, last bin, and number of bins
    FstBin = int(pOneRadial->RangeToFirstGateOfRef/pOneRadial->GateSizeOfReflectivity+0.5);
    BinNum = pOneRadial->GatesNumberOfReflectivity;
    if(FstBin<0)
    {
        BinNum = FstBin+BinNum;
        FstBin = -1*FstBin;
    }
    LstBin = FstBin + BinNum;
    ptrPos = pOneRadial->PtrOfReflectivity;

    //Save data
    for(BnIndex=FstBin; BnIndex<LstBin; BnIndex++)
        RData[EIIndex][AzIndex][BnIndex] = DecodeRef(pOneRadial->Echodata[ptrPos+BnIndex]);
}

//Save velocity data into the array
if(VFlag)
{
    //Get first bin, last bin, and number of bins
    FstBin = int(pOneRadial->RangeToFirstGateOfDop/pOneRadial->GateSizeOfDoppler+0.5);
    BinNum = pOneRadial->GatesNumberOfDoppler;
    if(FstBin<0)
    {
        BinNum = FstBin+BinNum;
        FstBin = -1*FstBin;
    }
}

```

```

    LstBin = FstBin + BinNum;
    ptrPos = pOneRadial->PtrOfVelocity;
    //Save data
    for(BnIndex=FstBin; BnIndex<LstBin; BnIndex++)
        VData[ElIndex][AzIndex][BnIndex] =
            DecodeVel(pOneRadial->Echodata[ptrPos+BnIndex], pOneRadial->ResolutionOfVelocity);
    }

    //Save spectrum width data into the array
    if(WFlag)
    {
        //Get first bin, last bin, and number of bins
        FstBin = int(pOneRadial->RangeToFirstGateOfDop/pOneRadial->GateSizeOfDoppler+0.5);
        BinNum = pOneRadial->GatesNumberOfDoppler;
        if(FstBin<0)
        {
            BinNum = FstBin+BinNum;
            FstBin = -1*FstBin;
        }
        LstBin = FstBin + BinNum;
        ptrPos = pOneRadial->PtrOfSpectrumWidth;
        //Save data
        for(BnIndex=FstBin; BnIndex<LstBin; BnIndex++)
            WData[ElIndex][AzIndex][BnIndex] = DecodeSpw(pOneRadial->Echodata[ptrPos+BnIndex]);
    }
}while(fileEndFlag==0 && !VolEnd && !(readSize<1));

if(!VolEnd && VolBeg)
{
    fclose(fp);
    free(pOneRadial);
    printf("Error! Incomplete Volume Scan\\n");
    // TRACE("Error! Incomplete Volume Scan\\n");
    return false;
}

NumValidCuts = ElIndex+1;
fclose(fp);
free(pOneRadial);
return true;
}

//*****//
/* 名称: DecodeRef(unsigned char code)

```

```

/** 类型: 实型, 返回解码后的反射率 (DBZ) 。
/** 功能: 将读出的反射率数据解码。
/** 参数: code, 无符号的 BYTE 型, 读出的反射率编码值。
//*****//
float DecodeRef(unsigned char code)
{
if(code==CODE_INVALID) return VALUE_INVALID;
else if(code==CODE_RANFOLD) return VALUE_RANFOLD;
else
return (float((code-2.)/2.-32.5));
}

//*****//
/** 名称: DecodeVel(unsigned char code)
/** 类型: 实型, 返回解码后的径向速度 (M/S) 。
/** 功能: 将读出的径向速度数据解码。
/** 参数: code, 无符号的 BYTE 型, 读出的径向速度编码值;
/**      ResType, 2 字节整型, 速度精度标记, 精度不同, 解码方式不同
/**      ResType = 2: 精度为 0.5 M/S
/**      ResType = 4: 精度为 1.0 M/S
//*****//
float DecodeVel(unsigned char code, short ResType)
{
if(code==CODE_INVALID) return VALUE_INVALID;
else if(code==CODE_RANFOLD) return VALUE_RANFOLD;
else
{
if(ResType==RES_POINT_FIVE) //0.5 m/s
return (float((code-2.)/2.-63.5));
else
return (float((code-2)-127.));
}
}

//*****//
/** 名称: DecodeSpw(unsigned char code)
/** 类型: 实型, 返回解码后的谱宽 (M/S) 。
/** 功能: 将读出的谱宽数据解码。
/** 参数: code, 无符号的 BYTE 型, 读出的谱宽编码值。
//*****//
float DecodeSpw(unsigned char code)
{
if(code==CODE_INVALID) return VALUE_INVALID;
else if(code==CODE_RANFOLD) return VALUE_RANFOLD;

```

```

else
    return (float((code-2.)/2.-63.5));
}

//*****//
/* 名称: SavedataIntoFiles()
/* 类型: 布尔型, 保存数组到文件中的操作如果出错, 返回 FALSE。
/* 功能: 将反射率数组, 速度数组, 谱宽数组, 仰角数组分别保存到 3 个文件中。
/*      RefArray.dat 中保存仰角数, 仰角, 和体扫反射率数组的数据
/*      VelArray.dat 中保存仰角数, 仰角, 和体扫速度数组的数据
/*      spwArray.dat 中保存仰角数, 仰角, 和体扫谱宽数组的数据
/*      数据以二进制的方式保存, 存放顺序:
/*      1) 当前仰角层数, 类型为 4 字节整型, 长度为 4 字节
/*      2) 有效层数的仰角, 类型为 4 字节实型, 长度为 4*仰角层数 字节
/*      3) R, V 或 W 体扫数据, 类型为 4 字节实型, 长度为 4*MaxRads*Gates**仰角层数 字节
/* 参数: code, 无符号的 BYTE 型, 读出的谱宽编码值。
//*****//
bool SavedataIntoFiles()
{
FILE *fpR=0, *fpV=0, *fpW=0;
char fileR[] = "RefArray.dat";
char fileV[] = "VelArray.dat";
char fileW[] = "spwArray.dat";

//Open Ref. File for Saving
fpR = fopen(fileR, "wb");
if(fpR==0) return false;
//Save Ref. Array
fwrite(&NumValidCuts, sizeof(NumValidCuts), 1, fpR);
fwrite(Elevation, sizeof(float), NumValidCuts, fpR);
fwrite(RData, sizeof(float), NumValidCuts*MaxRads*RGates, fpR);
fclose(fpR);

//Open Ref. File for Saving
fpV = fopen(fileV, "wb");
if(fpV==0) return false;
//Save Ref. Array
fwrite(&NumValidCuts, sizeof(NumValidCuts), 1, fpV);
fwrite(Elevation, sizeof(float), NumValidCuts, fpV);
fwrite(VData, sizeof(float), NumValidCuts*MaxRads*VGates, fpV);
fclose(fpV);

//Open Ref. File for Saving
fpW = fopen(fileW, "wb");

```

```

if(fpW==0) return false;
//Save Ref. Array
fwrite(&NumValidCuts, sizeof(NumValidCuts), 1, fpW);
fwrite(Elevation, sizeof(float), NumValidCuts, fpW);
fwrite(WData, sizeof(float), NumValidCuts*MaxRads*WGates, fpW);
fclose(fpW);

return true;
}

```

```

-- 作者: newman
-- 发布时间: 2005-8-23 9:51:31

```

```

--

//DataBase.h
//SA, SB 雷达的距离库数
/**/

#define RGates 460 //S BAND 反射率距离库数
#define VGates 920 //S BAND 速度距离库数
#define WGates 920 //S BAND 谱宽距离库数
/**/

//CB 雷达的距离库数
/*

#define RGates 800 //C BAND 反射率距离库数
#define VGates 1600 //C BAND 速度距离库数
#define WGates 1600 //C BAND 谱宽距离库数
*/

#define MaxCuts 20 //最大仰角层数
#define MaxRads 360 //每层仰角上的方位数, 每度保留一个径向

#define CODE_INVALID 0 //编码值中的特殊标记, 表示无有效观测数据
#define CODE_RANFOLD 1 //编码值中的特殊标记, 表示有距离模糊

#define VALUE_INVALID -999. //实际值中的特殊标记, 表示无有效观测数据
#define VALUE_RANFOLD 999. //实际值中的特殊标记, 表示有距离模糊

#define RES_POINT_FIVE 2 //速度精度类型, 代表的精度为 0.5 M/S
#define RES_ONE_POINT 4 //速度精度类型, 代表的精度为 1.0 M/S

#define VOL_BEG 3 //体扫开始状态标志
#define VOL_END 4 //体扫结束状态标志

```

```

#define ELV_BEG 0    //仰角开始状态标志
#define ELV_END 2    //仰角结束状态标志

#define RADIAN 3.14159/180.
////////////////////////////////////
//tagBaseData 98D 雷达信息结构(目标结构)
typedef struct tagBaseData
{
    unsigned short  temp1[7];        //保留
    unsigned short  RadarStatus;     //1 - 表示为雷达数据
    unsigned short  temp2[6];        //保留
    unsigned int    mSeconds;         //径向数据收集时间
    unsigned short  JulianDate;       //从 1970/1/1 起的日期
    unsigned short  URange;           //不模糊距离
    unsigned short  Az;               //方位角度
    unsigned short  RadialNumber;      //径向数据序号
    unsigned short  RadialStatus;     //径向数据状态
    unsigned short  El;               //仰角
    unsigned short  ElNumber;         //体扫内的仰角编号
    short           RangeToFirstGateOfRef; //第一个反射率数据表示的实际距离(m)
    short           RangeToFirstGateOfDop; //第一个多普勒数据表示的实际距离(m)
    unsigned short  GateSizeOfReflectivity; //反射率数据的距离库长(m)
    unsigned short  GateSizeOfDoppler;     //多普勒数据的距离库长(m)
    unsigned short  GatesNumberOfReflectivity; //反射率数据的距离库数
    unsigned short  GatesNumberOfDoppler;    //多普勒数据的距离库数
    unsigned short  CutSectorNumber;        //扇区号
    unsigned int    CalibrationConst;       //标定常数
    unsigned short  PtrOfReflectivity;      //反射率数据指针
    unsigned short  PtrOfVelocity;         //速度数据指针
    unsigned short  PtrOfSpectrumWidth;    //谱宽数据指针
    unsigned short  ResolutionOfVelocity;  //多普勒速度分辨率
    unsigned short  VcpNumber;             //体扫号
    unsigned short  temp4[4];              //保留
    unsigned short  PtrOfArcReflectivity;   //反射率数据指针
    unsigned short  PtrOfArcVelocity;       //速度数据指针
    unsigned short  PtrOfArcWidth;         //谱宽数据指针
    unsigned short  Nyquist;               //不模糊速度
    unsigned short  temp46;                //保留
    unsigned short  temp47;                //保留
    unsigned short  temp48;                //保留
    unsigned short  CircleTotal;           //仰角数
    unsigned char  temp5[30];              //保留
    unsigned char  Echodata[RGates+VGates+WGates]; //129—588 共 460 字节反射率数据

```

```

unsigned char temp[4];      //保留
                        //129—1508 共 1380 字节速度数据
                        //129—2428 共 2300 字节谱宽数据
}RADIALDATA;

float RData[MaxCuts][MaxRads][RGates];
float VData[MaxCuts][MaxRads][VGates];
float WData[MaxCuts][MaxRads][WGates];
float Elevation[MaxCuts];

// void DispSpw(CDC * pDC, CPoint cp, int r);
// void DispVel(CDC * pDC, CPoint cp, int r);
// void DispRef(CDC *pDC, CPoint cp, int r);
int NumValidCuts;
bool SavedataIntoFiles();
bool ReadBaseData(char filename[80]);
RADIALDATA* pOneRadial;

float DecodeSpw(unsigned char code);
float DecodeVel(unsigned char code, short ResType);
float DecodeRef(unsigned char code);

```