SWAN产品格式整理

1. D131格点数据格式

131格式经过数次修订，目前大多数数据仍未1.0格式，但是显示引擎已经能够解析2.0格式的数据，具体应用可以根据情况使用相应版本的格式，主要是从后面的Reserved中修改为对应版本的大小。保证文件头格式总长度1024个字节。

char ZonName[12]; // diamond 131 12个字节

char DataName[38];//数据说明(例如 2008年5月19日雷达三维拼图)38个字节

char Flag[8]; // 文件标志，"swan"

char Version[8]; // 数据版本号，"2.0" 2.0表示为版本为2.0

unsigned short int year;//2008 两个字节

unsigned short int month;//05 两个字节

unsigned short int day;//19 两个字节

unsigned short int hour;//14 两个字节

unsigned short int minute;//31 两个字节

unsigned short int interval ; //两个字节

unsigned short int XNumGrids;//1300 两个字节

unsigned short int YNumGrids;//800 两个字节

unsigned short int ZNumGrids;//20 两个字节

int RadarCount; //拼图雷达数 四个字节

float StartLon; //网格开始经度（左上角） 四个字节

float StartLat; //网格开始纬度（左上角） 四个字节

float CenterLon;//网格中心经度 四个字节

float CenterLat;//网格中心纬度 四个字节

float XReso; //经度方向分辨率 四个字节

float YReso; //纬度方向分辨率 四个字节

float ZhighGrids[40];//垂直方向的高度（单位km）数目根据ZnumGrids而得（最大40层） 160个字节。

char RadarStationName[20][16]; //相关站点名称, 20\*16字节

float RadarLongitude[20]; //相关站点所在经度，单位：度， 4\*20字节

float RadarLatitude[20]; //相关站点所在纬度，单位：度， 4\*20字节

float RadarAltitude[20]; //相关站点所在海拔高度，单位：米， 4\*20字节

unsigned char MosaicFlag[20]; //该相关站点数据是否包含在本次拼图中，未包含:0，包含:1, 20字节

short m\_iDataType; //0-unsigned char 1-char 2-unsigned short 3-short 4

//每一层的向量数，版本=2.0

short m\_iLevelDimension;

float offset;

float scale

char reserved[160] 保持头有1024个字节

接下来是数据块，从底层到高层进行排列共ZnumGrids层。每层的数据从起始点（左上角）开始，按维向（纬度y）减小写每行的经向（经度x增大）数据。数据存储按照m\_iDataTyp表示，分为不同的数据类型，存储值和真实值之间的换算关系为 data（真实值） =scale\* value(存储值)+ offset

1. D35 矢量数据格式
2. 标准格式

D35文件头包含D35标示，版本号（作为扩展格式的标示），基本信息（时间、记录数、无效值），表格定义（可分为带字段类型的TABDEF和普通的TAB），元数据属性，排列方式如下：

Diamond 35 文字描述

VERSION 1.0

文件生成时间 记录条数 无效记录表达值

TAB

表格描述

或

TAB 字段个数

表格描述

或者

TABDEF 字段个数

表格描述和字段类型，如果 字段一：Int 字段二：float 字段三：String

使用空格作为属性对的分割

PROPERTY

字段名：属性 字段名：属性 字段名：属性

DATA

记录类型 记录坐标点数

经度 纬度 高度

经度 纬度 高度

经度 纬度 高度

.....

字段1 字段2 字段3 .......

1. 简化格式

为了适应计算的简易化要求，在标准格式外还定义了一种一种简化格式。

文本格式，文件后缀TXT：

Diamond 35 文字描述

TAB（类型描述符） 文件生成时间 记录条数 无效记录表达值

记录字段数 字段1名 字段2名 字段3名 .......

记录类型 记录坐标点数

经度 纬度 高度

经度 纬度 高度

经度 纬度 高度

.....

字段1 字段2 字段3 .......

其中记录类型分为：POINT（点），MULTIPOINT（多个点），PLINE（直线），POLYGON（多边形）,POLYLINE(线条)

TAB下一行必须是表格定义

1. 自动站时序数据格式

自动站时序数据是二进制格式，包含了若干时长的自动站数据，文件首先包含文件头：

typedef struct tagSerialFile15

{

char szFlag[16];//标记=SwanCycle

short nVer;//版本=15 2011年月日从Int改为Short型

int nDataOffset;//数据区开始的位置,从文件头开始计算

short nIndex;//2011年月日增加了最新记录的索引号

char szRecordTime[16];//记录时间

int nRecordInterval;//标准的观测间隔

int nTotalTime;//记录时长

int nTotalSta;//站点的个数

int nRecordSize;//每个站点记录长度

} SERIALFILE15;

然后从nDataOffset的位置开始存放自动站的站点数据，每个站点的数据长度为nRecordSize。每个站点保存了站点数据，每个数据指针包含了若干个时次的数据，具体个数由nTotalTime和nTotalTime决定，结构如下：

typedef struct tagAwsInfoSerial

{

STAINFO sta;

float\* fWindDir\_2Min;//2分钟平均风向

float\* fWindSpd\_2Min;//2分钟平均风速

float\* fWindDir\_10Min;//10分钟平均风向

float\* fWindSpd\_10Min;//10分钟平均风速

float\* fWindDir\_Max;//最大风向

float\* fWindSpd\_Max;//最大风速

char\* szWindTime\_Max;//最大风速时间,以8个字节为一段

float\* fRain;

float\* fRain\_PerMin;//分钟雨量，申请数为保留小时\*60

float\* fTemp;//温度

float\* fTempHigh;//最高温度(1小时内的)

float\* fTempLow;//最低温度（小时内的）

float\* fTempLd;//露点温度

float\* fXdsd;//相对湿度

float\* fQy;//气压

float\* fNjd;//能见度

} AWSINFOSERIAL;

其中站点信息的子结构定义如下：

typedef struct tagStaInfo

{

int nStaId; //站号

char szStaName[32];//站名

char szCity[32];//所属地市名

float fLon;//经度

float fLat;//纬度

float fHei;//高度

int nLevel;//级别

} STAINFO;

1. TITAN数据格式

首先是文件头标志，8个字节，“NMCTITAN”；

然后是4个字节，版本号，整型数，目前为1；

其次预留16字节，以后备用，目前为空；

再次是当前追踪风暴的时间信息(世界时)，共8个字节，依次为年月日时分，各占用2个字节：如，

unsigned short Year;

unsigned short Month;

unsigned char Day;

unsigned char Hour;

unsigned char Minute;

unsigned char Second; ///当前时间，世界时

接下来是4个字节，整型数，表示过去时次与当前时次追踪风暴的时次数nTrackTimeCount；（目前追踪过去1小时，6分钟一次，共11时次）

接下来为nTrackTimeCount个以下数据结构：

typedef struct {

unsigned short Year;

unsigned short Month;

unsigned char Day;

unsigned char Hour;

unsigned char Minute;

unsigned char Second; ///当前时间，世界时

int nStormCount; // 4个字节，整型数，表示风暴的个数；

TITAN\_STORM storm[nStormCount];//风暴信息

storm\_props props[nStormCount]; //风暴属性

}TitanStormTrack;

然后是4个字节，整型数，表示预报风暴的时次数nForecastCount；（目前预报1小时，6分钟一次，共10时次）。

接下来为nForecastCount个以下数据结构：

typedef struct {

unsigned short nForecastMinute; ///预报时间，单位分钟

int nStormCount; // 4个字节，整型数，表示风暴的个数；

TITAN\_STORM storm[nStormCount];//风暴信息

}TitanStormForecast;

其中：

typedef struct

{

double proj\_area\_centroid\_x; //风暴投影中心经度

double proj\_area\_centroid\_y; //风暴投影中心纬度

double vol\_centroid\_z; //风暴中心高度

double refl\_centroid\_z; //风暴反射率因子中心高度

double top; //风暴顶高度

double dbz\_max; //最大反射率因子

double volume; //风暴体积

double mass; //风暴质量

double proj\_area; //风暴面积

double smoothed\_speed;//风暴移动速度

double smoothed\_direction;//风暴移动方向，正北为0

double vil; //风暴VIL值

} storm\_props; //风暴属性

typedef struct{

int nStormBoundaryPointNum;//风暴边界点个数

FPOINT lonlat[nStormBoundaryPointNum]; // 风暴边界点

}TITAN\_STORM;

typedef struct{

float lon; //经度

float lat; //纬度

}FPOINT;

1. SICT数据格式

风暴追踪和预报的格式，参照了WSR－88D SCIT格式，但做了少量的调整，具体如下：

首先是文件头标志，8个字节，“GZ\_STORM”；

然后是4个字节，版本号，整型数，目前为1；

其次预留16字节，以后备用，目前为空；

再次是当前追踪风暴的时间信息(世界时)，共10个字节，依次为年月日时分，各占用2个字节：如，

unsigned short Year;

unsigned short Month;

unsigned short Day;

unsigned short Hour;

unsigned short Minute; ///当前时间，世界时

接下来4个字节，整型数，表示风暴的个数；

其次是如下的一个结构，有多少个风暴，就有多少个下面的结构

typedef struct { //包括过去一小时轨迹，当前时刻，以及未来一个小时的预报

PRE\_TIME pre\_time[10]; ////0－9表示前一到十个时刻

POINT\_XY current\_time; ///对于当前时刻和未来的十个时刻，记录风暴位置

POINT\_XY next\_time[10]; ///0－9表示未来的一到十个时刻

int Number; ///组成该风暴的点数，针对当前时刻而言

int Radius; //该风暴的半径，针对当前时刻而言

int ID; //风暴的ID，同时时次独一无二，介于[1,999],循环使用

unsigned short Start\_Year;

unsigned short Start\_Month;

unsigned short Start\_Day;

unsigned short Start\_Hour;

unsigned short Start\_Minute; ///风暴的生命开始的时间，世界时

}StormTracking;

其中：

////定义历史轨迹

typedef struct {

unsigned char Symbol; ///\*等于0，1，2分别表示“没有有效信息”、“只有一个位置信息（包括新生、分裂、一对一匹配）、“有两个位置信息（合并）”\*/

POINT\_XY mainpoint; ///\* 单体“新生”、“分裂”和“一对一匹配”时的位置，此时是唯一；“合并”时，最多允许两个合并，这是主风暴位置\*/

POINT\_XY addpoint; ///\*附加的一个additional，当出现“合并”时，除上述主风暴外，还有这个附加的\*/

}PRE\_TIME;

typedef struct{

short x; //除以100，即为度数，如11336表示113.36度

short y; //除以100，即为度数，如2336表示23.36度

}POINT\_XY; ///由于目前的雷达拼图只是水平方向上的二维平面，没有Z