

About GrADS Station Data

This section describes the structure of station data files, how to create them, and how to instruct GrADS to interpret them properly. Please refer to the companion section on [Using Station Data](#) for information about the GrADS commands and functions that are available for analyzing and displaying station data. Here are some quick links for skipping through this section:

- [Structure of a Station Data File](#)
 - [Creating a Station Data File](#)
 - [Station Data Descriptor File](#)
 - [The STNMAP Utility](#)
-

Structure of a Station Data File

Station data are written to a binary file one report at a time. Groups of station reports are ordered within the file according to the time interval. The time interval for a set of upper air soundings might be 12 hours, and the time interval for a set of surface observations might be 1 hour.

Variables within each report are split into two categories: surface and level-dependent. Surface variables are reported at most once per report, and level-dependent variables are reported at multiple pressure levels within each report.

Each station report in a binary station data file has the following structure:

- A header which provides information about the location of the station
- Surface variables, if any
- Level dependent variables, if any

The header is described by the following C language data structure:

```
struct reporthheader {
    char id[8];      /* Station ID */
    float lat;      /* Latitude of Station */
    float lon;      /* Longitude of Station */
    float t;        /* Time in grid-relative units */
    int nlev;       /* Number of data groups following the header
*/
    int flag;       /* Level independent var set flag */
};
```

A detailed description of each header entry follows:

id

The station ID uniquely identifies the station. It can be 1 to 7 characters long and may be assigned arbitrarily; ie. the stations could be numbered in some arbitrary order.

lat, lon

The location of the station, given in world coordinates (latitude and longitude).

t

The time of this report, in grid-relative units. This refers to the way the stations are grouped in time. For example, if you are working with surface airways reports, you would probably have a time grouping interval of one hour. If you wanted to treat the report times of each report as being exactly on the hour, you would set t to 0.0. If the report was for 12:15pm, and you were writing the time group for 12pm, you would set t to be 0.25. Thus, t would typically have the range of - 0.5 to 0.5.

nlev

Number of data groups following the header. This is the count of the one surface group, if present, plus the number of level dependent groups. Is set to zero to mark the end of a time group in the file.

flag

If set to 0, there are no surface variables following the header. If set to 1, then there are surface variables following the header.

The surface variable data (if present) are written to file following the header. Surface variables are written out as floating point numbers in the order they are listed in the data descriptor file. Each of the surface variables must be written -- missing variables should contain the missing data value specified in the data descriptor file. The group of surface variable data must be the same size for each report in the file.

The level-dependent variables are written to file following the surface variables as follows:

level -- a floating point value giving the Z dimension in world coordinates for this level.

variables -- The level-dependent variables for this level.

Each level dependent group must have all the level dependent variables present, even if they are filled with the missing data value. The group of level dependent variable data must be the same size for all levels and all reports in the file.

After all the reports for one time grouping have been written, a special header (with no data groups -- nlev set to zero) is written to indicate the end of the time group. The next time group may then start immediately after. A time group with no reports would still contain the time group terminator header record (ie, two terminators in a row).

Creating a Station Data File

GrADS station data files must be written out in the structure outlined in the previous section. Examples of C and FORTRAN programs to create station data sets are provided below.

Let's say you have a data set with monthly rainfall:

Year	Month	Stid	Lat	Lon	Rainfall
1980	1	QQQ	34.3	-85.5	123.3
1980	1	RRR	44.2	-84.5	87.1
1980	1	SSS	22.4	-83.5	412.8
1980	1	TTT	33.4	-82.5	23.3
1980	2	QQQ	34.3	-85.5	145.1
1980	2	RRR	44.2	-84.5	871.4
1980	2	SSS	22.4	-83.5	223.1
1980	2	TTT	33.4	-82.5	45.5

A sample DEC FORTRAN program to write this data set in GrADS format is given below. Note that the OPEN statement is set to write a stream data set. This option may not be available with every compiler. If your program writes out data in sequential format, you must add an "OPTIONS sequential" entry to your GrADS data descriptor file.

```

CHARACTER*8 STID
OPEN (8,NAME='rain.ch')
OPEN (10,NAME='rain.dat',FORM='UNFORMATTED',RECORDTYPE='STREAM')
IFLAG = 0
C Read and Write
10 READ (8,9000,END=90) IYEAR,IMONTH,STID,RLAT,RLON,RVAL
9000 FORMAT (I4,3X,I2,2X,A8,3F8.1)
IF (IFLAG.EQ.0) THEN
    IFLAG = 1
    IYOLD = IYEAR
    IMNOLD = IMONTH
ENDIF
C If new time group, write time group terminator.
C Assuming no empty time groups.
IF (IYOLD.NE.IYEAR.OR.IMNOLD.NE.IMONTH) THEN
    NLEV = 0
    WRITE (10) STID,RLAT,RLON,TIM,NLEV,NFLAG
ENDIF
IYOLD = IYEAR
IMNOLD = IMONTH
C Write this report
TIM = 0.0
NLEV = 1

```

```

        NFLAG = 1
        WRITE (10) STID,RLAT,RLON,TIM,NLEV,NFLAG
        WRITE (10) RVAL
        GO TO 10
C On end of file write last time group terminator.
90     CONTINUE
        NLEV = 0
        WRITE (10) STID,RLAT,RLON,TIM,NLEV,NFLAG
        STOP
        END

```

An equivalent C program might be:

```

#include <stdio.h>
/* Structure that describes a report header in a stn file */
struct rpthdr {
    char  id[8];      /* Station ID */
    float lat;       /* Latitude of Station */
    float lon;       /* Longitude of Station */
    float t;        /* Time in grid-relative units */
    int   nlev;     /* Number of levels following */
    int   flag;     /* Level independent var set flag */
} hdr;

main ()
{
    FILE  *ifile, *ofile;
    char  rec[80];
    int   flag,year,month,yrsav,mnsav,i;
    float val;

    /* Open files */
    ifile = fopen ("rain.ch","r");
    ofile = fopen ("rain.dat","wb");
    if (ifile==NULL || ofile==NULL) {
        printf("Error opening files\n");
        return;
    }

    /* Read, write loop */
    flag = 1;
    while (fgets(rec,79,ifile)!=NULL) {
        /* Format conversion */
        sscanf (rec,"%i %i",&year,&month);
        sscanf (rec+20," %g %g %g",&hdr.lat,&hdr.lon,&val);
        for (i=0; i<8; i++) hdr.id[i] = rec[i+11];
        /* Time group terminator if needed */
        if (flag) {
            yrsav = year;
            mnsav = month;
            flag = 0;
        }
        if (yrsav!=year || mnsav!=month) {
            hdr.nlev = 0;
            fwrite(&hdr,sizeof(struct rpthdr), 1, ofile);

```

```

    }
    yrsav = year;
    mnsav = month;
    /* Write this report */
    hdr.nlev = 1;
    hdr.flag = 1;
    hdr.t = 0.0;
    fwrite (&hdr,sizeof(struct rpthdr), 1, ofile);
    fwrite (&val,sizeof(float), 1, ofile);
}
hdr.nlev = 0;
fwrite (&hdr,sizeof(struct rpthdr), 1, ofile);
}

```

Station Data Descriptor File

After creating a binary file containing your station data, you must write a station data descriptor file so GrADS knows how to interpret the binary data file. The format for the data descriptor file for station data is similar to the format for a gridded data set, but there are a few differences as well as additional entries that are unique to station data descriptor files. These differences are outlined below. For further information on all the entries of a descriptor file, consult the section of the User's Guide on [Elements of a GrADS Data Descriptor File](#).

Here is an example of a station data descriptor file. Remember that the variables must be listed in the same order as they were written to the binary file.

```

DSET   ^station_data_sample.dat
DTYPE  station
STNMAP  station_data_sample.map
UNDEF  -999.0
TITLE  Station Data Sample
TDEF   10 linear 12z18jan1992 12hr
VARS  11
ps     0 99 Surface Pressure
ts     0 99 Surface Temperature
ds     0 99 Surface Dewpoint Temperature
us     0 99 Surface U-Wind
vs     0 99 Surface V-Wind
elev  0 99 Elevation of Station
z      1 99 Height
t      1 99 Temperature
d      1 99 Dewpoint Temperature
u      1 99 U-Wind
v      1 99 V-Wind
ENDVARS

```

Note the following differences between this descriptor file and a gridded data descriptor file:

```
DTYPE station
```

This entry identifies the data file as station data.

STNMAP *filename*

This entry identifies the file name of the station map file created by the [stnmap](#) utility.

XDEF, YDEF, and ZDEF

These entries are not included in a station data control file.

TDEF

This entry gives the number of the time groups in the file, the time stamp for the first group, and the interval between time groups.

VARs

The surface variables are listed first, and contain a "0" in the *levs* field. Level-dependent variables are listed after the surface variables, and contain a "1" in the *levs* field.

STNMAP Utility

Once the station data set has been created and the descriptor file has been written, the final step is to create the station map file by running the [stnmap](#) utility. This utility is executed externally from the command line, not from within GrADS. [stnmap](#) writes out information that allows GrADS to access the station report data more efficiently. The output from [stnmap](#) is called the station map file and its name is identified in the STNMAP entry of the data descriptor file. [stnmap](#) will prompt the user for the name of the data descriptor file, or it can be specified as an input argument on the command line. Station map files must be created on the machine where they are to be used. Consult the [reference page](#) for more information.

If you change the station data file, perhaps by appending another time group, then you will also have to change the descriptor file to reflect the changes and then rerun the [stnmap](#) utility.