

WXP File Reference

Version 5.0

12 August, 1998

Ingestor Files

Ingested File.....	1
Header File	3

Surface Files

ASCII Surface Data File.....	5
NetCDF Surface File.....	8

Upper Air Files

ASCII Upper Air File.....	10
NetCDF Upper Air File.....	15

Radar Files

ASCII MDR Radar File.....	18
NetCDF MDR Radar File.....	24
ASCII RCM Radar File.....	26
NIDS Radar File.....	30
NOWRad Radar File.....	31

Lightning Files

Albany NLDN File.....	32
Unisys NLDN File.....	33

Grid Files

ASCII/Binary Grid File.....	34
GRIB Grid File.....	37

Raw Files

ASCII Raw File.....	38
---------------------	----

Satellite/Image Files

McIDAS AREA File.....	43
NOAAPORT Satellite File.....	44
Unisys Satellite File.....	45
WXP Image File.....	47

Database Files

Bulletin File (.bul).....	49
Case Study Lookup File (case.lup).....	53
City Location File (.cty).....	55
Color Table File (.clr).....	56
Color Fill File (.cfl).....	57
Font File (.fnt).....	59
Font List File (.fnl).....	61
Map File (.map and .bmap).....	62
Map List File (.mpl).....	64
Model Lookup File (model.lup).....	66
Model Name File (mod_name.lup).....	68
Name Convention File.....	70
Parse Lookup File (parse.lup).....	72
Region File (.reg).....	73
Resource File (.wxpdef or wxp.def).....	74
Satellite Enhancement File (.enh).....	77
Shell Menu File (wxp.menu).....	79
Symbol File (.smb).....	81
Unit Conversion File (unit.lup).....	83
Variable File (.var).....	85
Variable Lookup File (variable.lup).....	94

Ingested File

The ingested data file contains the data that has been saved from the [ingest](#) program. This is used as the input to several WXP programs including decoders, parsers and some display programs.

FORMAT

The information is laid out in the file by product. These are limited as follows:

```

** header ***
product data
** header ***
.
.
.

```

where: *header* is the WMO header information describing the type of data.

The WXP ingestor strips control characters before saving to disk. To delimit the header, the header is surrounded by asterisks "***". The WXP ingestor can save binary (non-stripped) data. The header remains the same but the product content is the original binary data. Binary mode is triggered automatically if the ingestor sees GRIB or BUFR data. Otherwise, the "B" switch needs to be added to the bulletin file for those products that need to be saved in binary mode.

EXAMPLES

A sample of an **DD+** output file is:

```

** FPUS74 KOUN 022358 ***
NOWOKC

SHORT TERM FORECAST
NATIONAL WEATHER SERVICE NORMAN OK
656 PM CDT SUN AUG 2 1998

OKZ014-016-017-021>023-034-035-030100-
BECKHAM OK-BLAINE OK-CADDO OK-CUSTER OK-GREER OK-KIOWA OK-
ROGER MILLS OK-WASHITA OK-
INCLUDING THE CITIES OF -CLINTON/WEATHERFORD OK-ELK CITY OK-
HOBART OK-
656 PM CDT SUN AUG 2 1998

.NOW...
AN AREA OF SHOWERS OVER BECKHAM AND NORTHERN GREER COUNTIES WILL MOVE
EAST INTO WESTERN WASHITA AND KIOWA COUNTIES THROUGH 8 PM. BRIEF WIND
GUSTS TO NEAR 45 MPH WILL BE POSSIBLE IN THE VICINITY OF THE SHOWERS.
RAINFALL AMOUNTS WILL BE GENERALLY LESS THAN ONE-TENTH OF AN INCH.

$$

CJS

** FPUS83 KFGF 022358 ***

```

A sample of a **HDS** file is:

```

** YOQA10 KWBE 030000 ***[0A]
GRIB[00][0C],[01][00][00][1C][02][07]Y[D3][80]'d[00]db[08][03][00][00][01]
[00]
[00][00][00][00][00][14][00][00][00][00][00]*[00][FF][03][00]][00]A[00]/[9
E][82]
[09]S[08][81]s[18][01]=w[01]=w[00]@[00]a[A8][00]a[A8][00][00][00][00][00][
00][00]
[00][00][0B][DA][04][80][05][C0])w[04][04]fSwS#3Ee#FeD3ET3EVvwT4FvUVfU5fd
UUUDVFU
...
334EfffvfUTDEUUUEUVfUDUfUTUTEfUUEUUD3D33EfffUTDC4EVffgfeTDDUUUDDUUUUUV`777
7[0D]
[0D][0A]
[03][0A]
** YOQA15 KWBE 030000 ***[0A]
GRIB[00][0F]
[01][00][00][1C][02][07]Y[D3][80]'d[00][96]b[08][03][00][00][01][00]
[00][00][00][00][00][14][00][00][00][00][00]*[00][FF][03][00]][00]A[00]/[9
E][82]
[09]S[08][81]s[18][01]=w[01]=w[00]@[00]a[A8][00]a[A8][00][00][00][00][00][
00][00]
[00][00][0E][CE][07][80][05][C0]L[F8]2[05]BTel[89][08][08][83][9C][C7]1DB[
A4]
...

```

A sample **604** file is:

```

** 201 ***

SA EASTERN STATES 060000
MSS SA 2350 10 SCT E18 OVC 7 108/36/31/E0306/985
ART SA 2350 M18 OVC 15 111/35/30/0308/985/ 210 37
GFL SA 2353 E40 OVC 10 075/38/30/0112/973/ 217 41
ROC SA 2352 M22 OVC 15 132/38/32/3106/990/ 319 15// 39

```

SEE ALSO

- [ingest](#)

Header File

The header file contains the byte location of products saved by the [ingest](#) program. This is used as the input to several WXP programs including parsers and some display programs increase the speed of searching for various types of data.

FORMAT

The information is laid out in the file as:

```
000000 header / extra
000000 header / extra
....
```

where:

- *000000* -- the byte offset into the file,
- *header* -- the product header in its entirety is listed after the offset
- *extra* -- extra information which includes second line for text data and product information for GRIB data.

EXAMPLES

A sample from a forecast data header file:

```
0 FPUS86 KPQR 282359 / OPUPDX
3264 FPUS85 KGGW 290001 / OPUGGW
3548 FPAK11 PAYA 282207 / &ZCZC JNULFPYAK
4190 FPUS73 KFGF 282359 / NOWFAR
6865 FPAK57 PAJK 290001 CCA / &ZCZC JNUZFPAK
9613 FPUS73 KLBF 290002 / NOWLBF
10092 FPUS73 KGRB 290001 / NOWGRB
10588 FPAK11 PAFA 290005&ZCZC FAILFPFAI / AKZ007-290530-
11366 FCCN51 CWAO 290001 AAA / TAF AMD CYUY 290001Z
11592 FPAK57 PAJK 290001 CCA / &ZCZC JNUZFPAK
14342 FPAK11 PAFA 290005 / &ZCZC FAILFPFAI
```

The information after the slash "/" is the second line of the product used for further parsing. This generally contains AFOS PILs which can be used for parsing.

For GRIB products:

```
3087563 YORB10 KWBE 131500 / 85 212      6 100      100 39
3102432 ZORE10 KWBE 131500 / 85 212      9 100      100 39
3117301 YCUA99 KWBE 131500 PAA / 85 215      0 100     1000 41
3230510 YCUA85 KWBE 131500 PAA / 85 215      0 100      850 41
3336000 YCUA70 KWBE 131500 PAA / 85 215      0 100      700 41
3352511 YCUA50 KWBE 131500 PAA / 85 215      0 100      500 41
3368533 YCUA25 KWBE 131500 PAA / 85 215      0 100      250 41
3474022 YTUA98 KWBE 131500 PAA / 85 215      0 105        2 11
3487852 YUUA98 KWBE 131500 PAA / 85 215      0 105        10 33
3583433 YVUA98 KWBE 131500 PAA / 85 215      0 105        10 34
3599668 YRUA98 KWBE 131500 PAA / 85 215      0 105         2 52
3635495 YPUA98 KWBE 131500 PAA / 85 215      0  1         0  1
```

The extra characters after the slash for GRIB data is product information derived from the GRIB product definition block (PDB). This includes 6 numbers which represent:

1. Model number
2. Grid number (used to specify grid size and domain)
3. Forecast time (usually in hours, coded for hour ranges)
4. Vertical level type
5. Vertical level
6. Variable number

To decode these numbers, see the [Appendix \(WXP Product Descriptions\)](#). Since GRIB headers are not unique, this information is needed to uniquely describe the contents of the product.

SEE ALSO

- [ingest](#)

ASCII Surface Data File

This type of file is generated by the surface METAR/SAO decoder (**sacvt**) and the synoptic decoder (**smcvt**) programs for use with other WXP surface display programs.

FORMAT

This is an ASCII format which makes the files editable by programs such as "vi" and "emacs". The format of the file is as follows:

```

WXPSFC
hhZ dd mmm yy
iii TTT ddd wwwww aaa ppp vvv hhhC[,hhhC...] WWW [@tttt] [LAAAA,OOOOO] [PTttt][QRRR] [rrrr][qrrr]
[RRRR] [Xxxx] [Nnnn] [Tee] [SSS] [Cccc] [Ggg] [ssss] [#ccccc...] $
iii TTT ddd wwwww aaa ppp vvv hhhC[,hhhC...] ... $
...

```

Header Format

The first line of the file is the string:

```
WXPSFC
```

which is used to determine file type. The second line of the data file contains the date and time in the following format:

```
hhnnZ dd mmm yy
```

Where:

- *hh* -- Hour of the observation in GMT.
- *nn* -- Minute of the observation (optional).
- *dd* -- Day of the observation.
- *mmm* -- A three letter abbreviation for the month.
- *yy* -- The last two digits of the year.

Example: **12Z 24 JUN 98**

Data Line Format

The data are entered on the following lines, one station per line. Each particular piece of information is separated by a space and the line is terminated with a space and a "\$". The format for each station is listed below:

```

iii TTT ddd wwwww aaa ppp vvv hhhC[,hhhC...] WWW [@tttt] [LAAAA,OOOOO] [PTttt][QRRR]
[rrrr][qrrr] [RRRR] [Xxxx] [Nnnn] [Tee] [SSS] [Cccc] [Ggg] [ssss] [#ccccc...] $

```

Where:

- *iii* -- Station identifier. If the station does not have an ICAO ID and the **id_enc** option is specified, an ID will be derived from the WMO number using the 2 letter country prefix and the last 3 numbers of the WMO number. Country prefixes are listed in the [Global Station Information Appendix](#).
- *TTT* -- Temperature in .1_F. Missing data is **-99**.
- *ddd* -- Dewpoint in .1_F. Missing data is **-99**.
- *wwwww* -- Wind data. If *wwwww* is 4 digits, the first two digits are the direction in 10's of degrees and the last two digits are the wind speed in knots. If *wwwww* > 5000, the first 3 digits are the wind direction in degrees and the last 3 digits are the wind speed. Missing winds are encoded **-999**.

- *aaa* -- Altimeter setting in .01 inches of Mercury. Only 3 digits are displayed. To convert to real altimeter setting: $30+aaa/100$ if *aaa* is less than 500, otherwise, $20+aaa/100$. If *aaaa* is a four digit number, the altimeter setting is *aaaa*/100. Missing data is **-99**.
- *ppp* -- Sea level pressure in .1 millibars. Only 3 digits are displayed. To convert to real altimeter setting: $1000+ppp/10$ if *ppp* is less than 500, otherwise, $900+ppp/10$. This can be modified by checking altimeter setting, if altimeter setting > 30.8" or temperature < -40_F, 1000 is added. If altimeter setting < 28.3, 900 is added. If *pppp* is a four or five digit number, the pressure is *pppp*/10. Missing data is **-99**.
- *vvv* -- Visibility in miles. Missing data is **-99**.
- *hhhC* -- Cloud base *hhh* in 100's of feet and cloud cover *C*. Multiple levels can be specified separated by commas. (**-99** for missing) Cloud cover can be:
C-clear, F - few, S-scattered, B-broken, O-overcast, X-obscured, M - missing, s, b, o, x-thin layers, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9-Eighths of sky coverage with 9 being obscure.
- *W* -- Present weather. This is the standard SAO weather string (i.e. **RW-F**).
If weather has a leading **!**, this is the METAR string (i.e. **-RA**).
If the weather is a number, it refers to the 2 digit international coding table.
Missing data is **'-'**;
- *@ttt* -- Observation time (*HHMM*). (OPTIONAL)
- *LAAAA,OOOOO* -- The latitude and longitude of the station in .1 degrees. (OPTIONAL)
- *PTtt* -- Pressure tendency in .1 millibars/three hours. The first digit "*T*" represents the type of fall or rise. (**0-3**:rise, **5-8**:fall). The remaining three digits represent the tendency. (OPTIONAL)
- *QQQQ* -- 3 hours precipitation in .01 inches. (OPTIONAL)
- *rrrr* -- 6 hour precipitation in .01 inches. (OPTIONAL)
- *qqqq* -- 12 hour precipitation in .01 inches. (OPTIONAL)
- *RRRR* -- 24 hour precipitation in .01 inches. (OPTIONAL)
- *Xxxx* -- Maximum temperature in F. (OPTIONAL)
- *xxxx* -- 6 hour maximum temperature in F. (OPTIONAL)
- *Nnnn* -- Minimum temperature in F. (OPTIONAL)
- *nnnn* -- 6 hour minimum temperature in F. (OPTIONAL)
- *Tee* -- Extreme temperature in F. (OBSOLETE - OPTIONAL)
- *Sss* -- Snow cover in inches. (OPTIONAL)
- *Cccc* -- Cloud types for low, medium and high clouds. (OPTIONAL)
- *Ggg* -- Wind gusts in knots. (OPTIONAL)
- *ssss* -- Solar radiation in minutes. (OPTIONAL)
- *ttt* -- Sea Surface temperature in _C. (OPTIONAL)
- *pppp* -- Sea wave period in seconds. (OPTIONAL)
- *hhhh* -- Sea wave height in meters. (OPTIONAL)
- *cccc...* -- Comments (up to 34 characters).(OPTIONAL)

EXAMPLES

Example of raw surface data:

```
KIND 091756Z 16012KT 2 1/2SM RA BR SCT040 OVC046 07/06 A3002 RMK
AO2 PRESFR SLP170 P0012 60017 T00670061 10072 20033 58036
$=
```

The above line would be decoded into the following format:

```
KIND 441 430 1612 2 170 2.5 40S,460 !RA_BR @1756 P8-36 r17 x45.0 n37.9
#AO2 PRESFR P0012 $
```

A sample surface converted file would look like:

```
WXPSFC
1800Z 9 MAR 97
```

```
WRN 122 -4 3223 -99 263 9.0 -99M M @1745 G 29 #M/ 7007 $
WZN -99 -99 3515 -99 -99 -99 -99M M @1747 G 20 #M/M 7018 $
WBK 104 -22 3215 -99 299 9.0 -99M M @1746 #7009 $
KHKY 608 374 0908 43 -99 15 50S,900 - @1752 P7-19 x61.5 n37.0 #SLPNO $
KMWC -99 -99 1907 988 -99 1.0 60 !-RA_BR @1745 $
KTRK 340 261 0000 42 -99 30 180S - @1745 n18.0 #NOSPECI $
KART 284 32 3105 51 340 15 -99C - @1748 P7-20 x29.5 n-4.7 $
...
```

FILES

- **syn.cty** -- used as cross reference between WMO number and ICAO ID.

SEE ALSO

- [sacvt](#)
- [smcvt](#)
- [sfcwx](#)
- [sfccalc](#)
- [sa_parse](#)
- [statlog](#)

NetCDF Surface File

This type of file is generated by the **sacvt** and **smcvt** programs for use with other WXP surface display programs. The netCDF data contains the same information as the WXP ASCII format.

FORMAT

The information is laid out in the file using the following CDL structure:

```
netcdf 07031197 {
dimensions:
    report = UNLIMITED ; // (1447 currently)
    time_len = 20 ;
    id_len = 12 ;
    layers = 4 ;
    remarks_len = 35 ;

variables:
    char id(report, id_len) ;
        id:long_name = "station id" ;
    char region(report, id_len) ;
        region:long_name = "region id" ;
    char time(report, time_len) ;
        time:long_name = "time" ;
        time:units = "text_time" ;
    float lat(report) ;
        lat:long_name = "latitude" ;
        lat:units = "degrees_N" ;
        lat:_FillValue = -9999.f ;
    float lon(report) ;
        lon:long_name = "longitude" ;
        lon:units = "degrees_E" ;
        lon:_FillValue = -9999.f ;
    float elev(report) ;
        elev:long_name = "elevation" ;
        elev:units = "meters" ;
        elev:_FillValue = -9999.f ;
    float T(report) ;
        T:long_name = "temperature" ;
        T:units = "celsius" ;
        T:_FillValue = -9999.f ;
    float TD(report) ;
        TD:long_name = "dew point" ;
        TD:units = "celsius" ;
        TD:_FillValue = -9999.f ;
    float PSL(report) ;
        PSL:long_name = "sealevel pressure" ;
        PSL:units = "hectopascals" ;
        PSL:_FillValue = -9999.f ;
    float ALTIM(report) ;
        ALTIM:long_name = "altimeter setting" ;
        ALTIM:units = "hectopascals" ;
        ALTIM:_FillValue = -9999.f ;
    float SPD(report) ;
        SPD:long_name = "wind speed" ;
        SPD:units = "meters/second" ;
```

```

        SPD:_FillValue = -9999.f ;
float DIR(report) ;
        DIR:long_name = "wind direction" ;
        DIR:units = "degrees" ;
        DIR:_FillValue = -9999.f ;
float GUST(report) ;
        GUST:long_name = "wind gusts" ;
        GUST:units = "meters/second" ;
        GUST:_FillValue = -9999.f ;
byte WX(report, layers) ;
        WX:long_name = "weather" ;
        WX:units = "WMO table 4677" ;
float ZCL(report, layers) ;
        ZCL:long_name = "cloudbase" ;
        ZCL:units = "meters" ;
        ZCL:_FillValue = -9999.f ;
char CC(report, layers) ;
        CC:long_name = "cloudcover" ;
        CC:units = "WMO table 2700" ;
char cloudtype(report, layers) ;
        cloudtype:long_name = "cloudtypes" ;
        cloudtype:units = "WMO tables 0509, 0513, 0515" ;
float VIS(report) ;
        VIS:long_name = "visibility" ;
        VIS:units = "kilometers" ;
        VIS:_FillValue = -9999.f ;
char remarks(report, remarks_len) ;

// global attributes:
        :title = "Surface converted data" ;
        :version = "2.0" ;
        :history = "Surface converted file from WXP decoders" ;
        :filetime = "0700Z 11 MAR 97" ;
}

```

SEE ALSO

- [sacvt](#)
- [smcvt](#)
- [sfcwx](#)
- [sfccalc](#)
- [sa_parse](#)
- [statlog](#)

ASCII Upper Air File

This type of file is generated by the **uacvt** program for use with other WXP upper air display programs.

FORMAT

The format of the file is as follows:

WXPUPAx

hhZ dd mmm yy

wwwww:iiii:aaaa:oooo 00HHH TTTt dddff 92HHH TTTt dddff 85HHH TTTt dddff 70HHH TTTt dddff

50HHH TTTt dddff 40HHH TTTt dddff 30HHH TTTt dddff 25HHH TTTt dddff

20HHH TTTt dddff 15HHH TTTt dddff 10HHH TTTt dddff 07HHH TTTt dddff

05HHH TTTt dddff 03HHH TTTt dddff 02HHH TTTt dddff 01HHH TTTt dddff

88ppp TTTt dddff 77ppp dddff

ppp TTTt ppp TTTt ...

ppp TTTt ppp TTTt ... X

hh dddff hh dddff ...

hh dddff hh dddff ... \$

Header Format

The first line of the file is the string:

WXPUPAx

which is used to determine file type. The second line of the data file contains the date and time in the following format:

hhZ dd mmm yy

Where:

- *hh* -- Hour of the observation in GMT
- *dd* -- Day of the observation
- *mmm* -- A three letter abbreviation for the month
- *yy* -- The last two digits of the year

Example: **12Z 24 JUN 98**

Data Format

The data are entered on the following lines. This format listed below contains twelve lines of data for each individual station, which are in the following format:

Line 1-5: Mandatory level data to 500 mb (from TTAA, TTCC, or PPAA)

[wwwww:]iiii[aaaa:oooo] 00HHH TTTt dddff 92HHH TTTt dddff 85HHH TTTt dddff 70HHH TTTt dddff

50HHH TTTt dddff 40HHH TTTt dddff 30HHH TTTt dddff 25HHH TTTt dddff

20HHH TTTt dddff 15HHH TTTt dddff 10HHH TTTt dddff 07HHH TTTt dddff

05HHH TTTt dddff 03HHH TTTt dddff 02HHH TTTt dddff 01HHH TTTt dddff

88ppp TTTt dddff 77ppp dddff

Where:

- *wwwww* -- WMO number for the observation site (optional).
- *iiii* -- ICAO Station identifier.

If the station does not have an ICAO ID, one will be derived from the WMO number using the 2 letter country prefix and the last 3 numbers of the WMO number. Country prefixes are listed in the [Global Station Information Appendix](#).

- *aaaa* -- Station latitude (optional).
- *oooo* -- Station longitude (optional).
- *PP* -- Pressure level (see below). The value "**88**" represents tropopause data. The value "**77**" represents maximum wind level data. In these cases pressure of the level is reported instead of height.
- *ppp* -- Pressure of level. (999 for missing)
- *HHH* -- Height in meters. (999 for missing)

PP	Pressure	Code(H)	Height	Code(H)	Height
00	1000 mb	H < 500	H	H >= 500	500-H
92	925 mb	all H	H		
85	850 mb	all H	H+1000		
70	700 mb	H < 500	H+3000	H >= 500	H+2000
50	500 mb	all H	H*10		
40	400 mb	all H	H*10		
30	300 mb	H < 500	H*10+10000	H >= 500	H*10
25	250 mb	H < 500	H*10+10000	H >= 500	H*10
20	200 mb	all H	H*10+10000		
15	150 mb	all H	H*10+10000		
10	100 mb	all H	H*10+10000		
07	70 mb	all H	H*10+10000		
05	50 mb	H < 500	H*10+20000	H >= 500	H*10+10000
03	30 mb	all H	H*10+20000		
02	20 mb	all H	H*10+20000		
01	10 mb	H < 500	H*10+30000	H >= 500	H*10+20000

- *TTT* -- Temperature at the pressure level in .1 C. (999 for missing)
If the tenths digit is odd, the temperature is negative. Otherwise, the temperature is positive. For example, 234 is 23.4C whereas 123 is -12.3C.
- *tt* -- Dewpoint depression in C. (99 for missing)
If *tt* is less than or equal to 50, then the dewpoint depression is in tenths of a C. Otherwise, *tt* is the dewpoint depression in C plus 50. For example, 65 gives a dewpoint depression of 15C, whereas 17 gives a depression of 1.7C.
- *ddd* -- Wind direction to the nearest 5 degrees (999 for missing)
- *ff* -- Wind speed in knots (99 for missing)
If *ddd* is not an even multiple of 5, the difference between *ddd* and an even multiple of 5 is then added to the wind speed as the hundredths digit. For example, 31523 (*dddff*) is 315_ at 23 knots whereas 25612 is 255_ at 112 knots.

If all the information for a particular group is missing, it is encoded as an "X".

Next Lines: Significant reporting levels (50 maximum from TTBB and TTDD reports)

Format:

ppp TTTtt ppp TTTtt ...

Where:

- *ppp* -- Pressure of significant level
If pressure is greater than 1000 mb, *ppp* is the pressure minus 1000.
- *TTTtt* -- is encoded same as TTAA data listed above.

Information terminated with "X"

Next Lines: Wind level information (50 maximum from PPBB and PPDD reports)

Format:

hh dddff hh dddff ...

Where:

- *hh* -- Height of the wind level is 1000s of feet.
- *dddff* -- is encoded as listed above

Information terminated with "\$"

EXAMPLES

An example of a converted report would be as follows:

```
72456:KTOP:39.07:-95.62 00121 X X 92806 22212 19012 85541 18618 24019
70189 10459 25042
50589 06164 26032 40760 16718 23537 30971 30758 24035 25097 41358 23538
20245 53557 24043 15425 64963 25515 10672 64566 28503 07889 62372 07014
05100 57575 05012 X X X X X X X X X
88147 65763 26016 X X
983 26639 947 23019 890 21628 839 17817 819 18257 788 15856 769 16058
678 08860 592 01501 548 04300 533 04911 526 06156 525 05959 510 05359
484 07565 479 08361 476 08548 448 11907 390 18309 359 21346 317 28550
286 33360 235 44760 216 49556 183 57559 133 66762 94 64967 92 65967
61 60174 45 56375 X
00 12007 01 12507 02 14511 03 22514 04 24018 06 23520 07 23523
08 24531 09 25537 10 25540 11 24545 12 24044 14 24538 16 26026
18 26526 20 25535 25 23537 26 23038 30 24037 35 23534 43 25544
44 26012 46 24513 49 27514 50 28008 54 29502 55 28503 62 06511
63 09011 70 05018 $
```

Where:

- **KTOP** is Topeka KS (WMO 72456, Lat=39.07, Lon=-95.62)

Mandatory level data (miss is missing data)

Code	Level	Pres (mb)	Hght (m)	Temp (C)	Depr (C)	W Dir (deg)	W Spd (knt)
00121 X X		1000	121	miss	miss	miss	miss
92806 22212 19012		925	806	22.2	1.2	190	12
85541 18618 24019		850	1541	18.6	1.8	240	19
70189 10459 25042		700	3189	10.4	9	250	42
50589 06164 26032		500	5890	-6.1	14	260	32
40760 16718 23537		400	7600	-16.7	1.8	235	37
30971 30758 24035		300	9710	-30.7	8	240	35
25097 41358 23538		250	10970	-41.3	8	235	38
20245 53557 24043		200	12450	-53.5	7	240	43
15425 64963 25515		150	14250	-64.9	13	255	15
10672 64566 28503		100	16720	-64.5	16	285	3
07889 62372 07014		70	18890	-62.3	22	70	14
05100 57575 05012		50	20100	-57.5	25	50	12
X X X		30	miss	miss	miss	miss	miss

X X X		20	miss	miss	miss	miss	miss
X X X		10	miss	miss	miss	miss	miss
88147 65763 26016	Trop	147		-65.7	13	260	16
X X	Max Wind	miss				miss	miss

Significant level data

Code	Level	Pres (mb)	Temp (C)	Depr (C)
983 26639	SFC	983	26.6	3.9
947 23019		947	23.0	1.9
890 21628		890	21.6	2.8
839 17817		839	17.8	1.7
819 18257		819	18.2	7
788 15856		788	15.8	6
769 16058		769	16.0	8
...				
133 66762		133	-66.7	12
94 64967		94	-64.9	17
92 65967		92	-65.9	17
61 60174		61	-60.1	24
45 56375		45	-56.3	25

Wind level data

Code	Level	Hght (ft)	W Dir (deg)	W Spd (knt)
00 12007	SFC	SFC	120	7
01 12507		1000	125	7
02 14511		2000	145	11
03 22514		3000	225	14
04 24018		4000	240	18
06 23520		6000	235	20
07 23523		7000	235	23
...				
54 29502		54000	295	2
55 28503		55000	285	3
62 06511		62000	65	11
63 09011		63000	90	11
70 05018		70000	50	18

A sample upper air converted file would look like:

```

WXPUPAx
0000Z  3 AUG 98
01001:ENJA:70.93:-8.67 00047 06612 15514 92686 04203 16518 85374 05256
17517 709
33 03567 15520
50552 16929 15020 40716 27923 17021 30916 43199 21019 25036 53999 22015
20178 48799 20012 15370 45199 22511 10640 45599 22510 07879 44399 20007
05104 44999 26506 03446 43199 00000 02722 39387 13514 01201 X X
88227 58399 22018 X X
005 06010 993 06806 894 03004 855 05250 779 01270 731 02365 644 06576
613 09758 585 12365 573 13562 558 15514 528 17507 522 16115 468 19727
393 29120 389 28949 368 31362 314 40799 239 56799 227 58399 214 58799

```

```
205 49599 193 47999 171 45399 127 44399 89 45999 33 44399 17 39787
15 34589 14 37188 11 36588 X
00 13509 $
01028:ENBJ:74.52:19.02 00088 05834 26510 92720 00400 29014 85397 00507
30522 709
40 06550 29015
...
```

FILES

- **upa.cty** -- used as cross reference between WMO number and ICAO ID.

SEE ALSO

- [uacvt](#)
- [upairwx](#)
- [uacalplt](#)
- [upcalc](#)
- [ua_parse](#)

NetCDF Upper Air File

This type of file is generated by the **uacvt** program for use with other WXP upper air display programs.

FORMAT

The netCDF data contains the same information as the WXP ASCII format. The information is laid out in the file using the following CDL structure:

```
netcdf 00031197 {
dimensions:
    report = UNLIMITED ; // (143 currently)
    mant_level = 20 ;
    sigt_level = 50 ;
    sigw_level = 50 ;
    string_len = 11 ;

variables:
    char id(report, string_len) ;
        id:long_name = "Station ID" ;
    char region(report, string_len) ;
        region:long_name = "Region ID" ;
    char type(report, string_len) ;
        type:long_name = "Report Origination" ;
    long idn(report) ;
        idn:long_name = "WMO Numeric Station ID" ;
    float lat(report) ;
        lat:long_name = "Latitude" ;
        lat:units = "degrees_N" ;
        lat:valid_range = -180.f, 180.f ;
        lat:_FillValue = -9999.f ;
    float lon(report) ;
        lon:long_name = "Longitude" ;
        lon:units = "degrees_E" ;
        lon:_FillValue = -9999.f ;
    long num_mant(report) ;
        num_mant:long_name = "Number of Mandatory Levels" ;
        num_mant:valid_range = 0, 20 ;
        num_mant:_FillValue = 0 ;
    long num_sigt(report) ;
        num_sigt:long_name = "Number of Significant Levels wrt T" ;
        num_sigt:valid_range = 0, 50 ;
        num_sigt:_FillValue = 0 ;
    long num_sigw(report) ;
        num_sigw:long_name = "Number of Significant Levels wrt W" ;
        num_sigw:valid_range = 0, 50 ;
        num_sigw:_FillValue = 0 ;
    float P_man(report, mant_level) ;
        P_man:long_name = "Pressure - Mandatory Levels" ;
        P_man:units = "hectopascals" ;
        P_man:valid_range = 0.f, 1500.f ;
        P_man:_FillValue = -9999.f ;
    float Z_man(report, mant_level) ;
        Z_man:long_name = "Geopotential - Mandatory Levels" ;
        Z_man:units = "meters" ;
        Z_man:valid_range = -150.f, 100000.f ;
```

```

        Z_man:_FillValue = -9999.f ;
float T_man(report, mant_level) ;
        T_man:long_name = "Temperature - Mandatory Levels" ;
        T_man:units = "celsius" ;
        T_man:valid_range = -100.f, 100.f ;
        T_man:_FillValue = -9999.f ;
float TD_man(report, mant_level) ;
        TD_man:long_name = "Dew Point - Mandatory Levels" ;
        TD_man:units = "celsius" ;
        TD_man:valid_range = -100.f, 100.f ;
        TD_man:_FillValue = -9999.f ;
float DIR_man(report, mant_level) ;
        DIR_man:long_name = "Wind Direction - Mandatory Levels" ;
        DIR_man:units = "degrees" ;
        DIR_man:valid_range = 0.f, 360.f ;
        DIR_man:_FillValue = -9999.f ;
float SPD_man(report, mant_level) ;
        SPD_man:long_name = "Wind Speed - Mandatory Levels" ;
        SPD_man:units = "meters/second" ;
        SPD_man:valid_range = 0.f, 300.f ;
        SPD_man:_FillValue = -9999.f ;
float P_sigt(report, sigt_level) ;
        P_sigt:long_name = "Pressure - Significant Levels wrt T" ;
        P_sigt:units = "hectopascals" ;
        P_sigt:valid_range = 0.f, 1500.f ;
        P_sigt:_FillValue = -9999.f ;
float T_sigt(report, sigt_level) ;
        T_sigt:long_name = "Temperature - Significant Levels wrt T" ;
        T_sigt:units = "celsius" ;
        T_sigt:valid_range = -100.f, 100.f ;
        T_sigt:_FillValue = -9999.f ;
float TD_sigt(report, sigt_level) ;
        TD_sigt:long_name = "Dew Point - Significant Levels wrt T" ;
        TD_sigt:units = "celsius" ;
        TD_sigt:valid_range = -100.f, 100.f ;
        TD_sigt:_FillValue = -9999.f ;
float Z_sigw(report, sigw_level) ;
        Z_sigw:long_name = "Altitude - Significant Levels wrt W" ;
        Z_sigw:units = "meters" ;
        Z_sigw:valid_range = -150.f, 100000.f ;
        Z_sigw:_FillValue = -9999.f ;
float DIR_sigw(report, sigw_level) ;
        DIR_sigw:long_name = "Wind Direction - Significant Levels wrt
W" ;
        DIR_sigw:units = "degrees" ;
        DIR_sigw:valid_range = 0.f, 360.f ;
        DIR_sigw:_FillValue = -9999.f ;
float SPD_sigw(report, sigw_level) ;
        SPD_sigw:long_name = "Wind Speed - Significant Levels wrt W" ;
        SPD_sigw:units = "meters/second" ;
        SPD_sigw:valid_range = 0.f, 300.f ;
        SPD_sigw:_FillValue = -9999.f ;

// global attributes:
        :title = "Upper Air Observations" ;
        :version = "2.0" ;
        :history = "Upper air converted file from WXP decoders" ;

```

```
:filetime = "0000Z 11 MAR 97" ;  
:ymmddhh = "1997031100" ;
```

```
}
```

FILES

- **upa.cty** -- used as cross reference between WMO number and ICAO ID.

SEE ALSO

- [uacvt](#)
- [upairwx](#)
- [uacalplt](#)
- [upcalc](#)
- [ua_parse](#)

ASCII MDR Radar File

This type of file is generated by the **radcv**t program for use with the WXP radar display program.

FORMAT

The format of the file is as follows:

```

WXPRAD
hhnnZ dd mmm yy
SDUS SUMMARY
+ rr ccc
echos
+ rr ccc
SDUS
+ rr ccc
...
SDXX STATIONS
iiii CCCC ppp tt TTT,dddrrr Mddff Mddff Mddff
iiii CCCC ...

```

Header Format

The first line of the file is the string:

```
WXPRAD
```

which is used to determine file type. The second line of the data file contains the date and time in the following format:

```
hhnnZ dd mmm yy
```

Where:

- *hhnn* -- Hour and minute of the observation in GMT
- *dd* -- Day of the observation
- *mmm* -- A three letter abbreviation for the month
- *yy* -- The last two digits of the year

Example: **21Z 14 JUN 98**

Radar Summary Data

The radar summary data are displayed on the following lines. Line three of the data file marks the start of the radar summary information:

```
SDUS SUMMARY
```

This is followed by a location line containing the following:

```
+ rr ccc
```

Where:

- *rr* -- The row of the "+" on a 90 row by 120 column grid.
- *ccc* -- The column of the "+".

The actual summary data are listed on the following lines up to the next location line. Each character on the following lines represents the radar echo value for a grid box starting in column *ccc* and incrementing one column for each character on the line. The line following the location line represents row *rr*+1, and each successive line increments the row by one. This sequence is repeated until the whole radar summary is displayed.

SDUS

+ rr ccc

echos

+ rr ccc

The last *rr ccc* sequence represents the physical row column location of that "+". An example of a summary listing is:

SDUS

+ 44 040

```

                1  2
                1  1  1
                21
                12  22
                12444
                24442
                23233
                332333
                13  24412
                3  1334  2
                222  2453  1
                2  465  2
                245
                3244
                12234442
                12225333
                2223243
                23545253
                22225553342
                14436623332
                2  1266  3432  4
                2  2  5552  212  64
                422  5533  24
                22  1  2554  4  3  25
                23211  6563  2  2
                233563642  35
                122522666566  4
                23155  4423  4  4
                1  2  22
                2

```

111

+ 44 080

Radar Site Data

After the radar summary is listed, decoded information from each MDR report is listed for radar annotation. A separator line is listed to denote the start of the station reports:

SDXX STATIONS

The MDR information is formatted as follows:

iiii CCCC ppp ttt TTT,ddrrr Mddff Mddff Mddff

Where:

- *iiii* -- Station identifier. (followed by a space)
- *CCCC* -- The precipitation configuration which may be: (followed by a space)
NA -- Not available
NE -- No echoes
OM -- Out for maintenance
AREA -- Area of precipitation
CELL -- Precipitation cell
LINE -- Line of precipitation
- *ppp* -- Precipitation type. ("*" for missing)
- *ttt* -- Precipitation trend which may be: ("*" for missing)
NEW -- New precipitation
NC -- No change in intensity
+ -- Increasing
- -- Decreasing
- *TTT,ddrrr*
Maximum precipitation tops and location. ("*" for missing)
- *TTT* -- Maximum precipitation tops in 100s feet.
- *ddd* -- Compass direction of top from station in degrees
- *rrr* -- Distance from station of top in nautical miles.
- *Mddff* -- Precipitation movement ("*" for missing)
M -- Type of precipitation which could be:
A -- Area of precipitation
C -- Precipitation cell
L -- Line of precipitation
- *dd* -- Direction of movement away from in degrees
- *ff* -- Speed of movement in knots.

An example of converted MDR reports are listed below:

```
SDXX STATIONS
MPX AREA RW++ * * * * *
EAX AREA RW++ * 540,199113 * * *
LSX AREA * * * * *
SGF AREA RW++ * 470,262085 * * *
GWX NA * * * * *
JAN AREA RW++ * * * * *
BLX NA * * * * *
GGW CELL * * * * *
MSX AREA RW+ * * * * *
```

```
TFX AREA * * 390,050067 * * *
LTX AREA RW++ * 190,084126 * * *
MHX AREA RW++ * 390,114086 C1006 * *
RAX NE * * * * *
BIS LN * * * C0911 * *
MBX NA * * * * *
MVX LN TRW++ * * * * *
LNX AREA * * * * *
OAX AREA RW++ * * * * *
```

Describing the information from MHX (Moorehead, NC). There is an AREA of very heavy rain showers (RW++). The maximum precipitation tops are 39,000 feet located 114 degrees at 86 nautical miles away from the radar site. The cell is moving from 100 degrees at 6 knots.

EXAMPLES

A sample radar converted file would look like:

```
WXPRAD
0030Z 3 AUG 98
SDUS SUMMARY
+ 01 081
...

                                     1
                                      4
                                     431
                                     2342

+ 44 081
SDUS
+ 43 081

                                     4444
                                     54522
                                    15442
                                     32
                                     32

                                     45
                                    252
                                     1 3
                                    21 3      5
                                     222
5                                     232
                                    22234
                                    55 55
4 22445      2
```

62
442
5
52
34
4

```

+ 90 081
SDUS
...
+ 90 002
SDXX STATIONS
BMX NE * * * * *
EOX NA * * * * *
HTX NA * * * * *
MOB CELL * * * * *
MXX NA * * * * *
LZK CELL * * * * *
SRX NA * * * * *
EMX CELL * * * * *
FSX AREA RW++ * * * * *
IWA AREA TRW++ * * * * *
...
MPX AREA RW++ * * * * *
EAX AREA RW++ * 540,199113 * * *
LSX AREA * * * * *
SGF AREA RW++ * 470,262085 * * *
GWX NA * * * * *
JAN AREA RW++ * * * * *
BLX NA * * * * *
GGW CELL * * * * *
MSX AREA RW+ * * * * *
TFX AREA * * 390,050067 * * *
LTX AREA RW++ * 190,084126 * * *
MHX AREA RW++ * 390,114086 C1006 * *
RAX NE * * * * *
BIS LN * * * C0911 * *
MBX NA * * * * *
MVX LN TRW++ * * * * *
LNX AREA * * * * *
OAX AREA RW++ * * * * *
...

```

FILES

- **rad.cty** -- used to located radar sites on the LFM grid

SEE ALSO

- [radcvt](#)
- [rad](#)

NetCDF MDR Radar File

This type of file is generated by the **radcvf** program for use with the WXP radar display programs.

FORMAT

The netCDF data contains the same information as the WXP ASCII format. The information is laid out in the file using the following CDL structure:

```
netcdf 06031197_rad {
dimensions:
    mdr_x = 120 ;
    mdr_y = 90 ;
    id_len = 12 ;
    radar_len = 12 ;
    time_len = 16 ;
    rpt_type_len = 4 ;
    maxcover = 3 ;
    maxprecip = 3 ;
    maxazran = 8 ;
    maxwidth = 4 ;
    maxmove = 4 ;
    maxheight = 8 ;
    report = UNLIMITED ; // (126 currently)

variables:
    char mdr_time(time_len) ;
        mdr_time:long_name = "MDR summary time" ;
        mdr_time:units = "YYYYMMDDHHMM UTC" ;
    float lon(mdr_x, mdr_y) ;
        lon:long_name = "longitude" ;
        lon:units = "degrees_east" ;
    float lat(mdr_x, mdr_y) ;
        lat:long_name = "latitude" ;
        lat:units = "degrees_north" ;
    byte mdr(mdr_x, mdr_y) ;
        mdr:long_name = "manually digitized radar summary" ;
    long rpt_date ;
        rpt_date:long_name = "report date" ;
        rpt_date:units = "YYYYDDMM UTC" ;
    char id(report, id_len) ;
        id:long_name = "station ID" ;
    long rpt_time(report) ;
        rpt_time:long_name = "report time" ;
        rpt_time:units = "HHMM UTC" ;
    char radar_type(report, radar_len) ;
        radar_type:long_name = "radar type" ;
    byte opstat(report) ;
        opstat:long_name = "operational status" ;
        opstat:units = "enumerated in rarep.h" ;
    char rpt_type(report, rpt_type_len) ;
        rpt_type:long_name = "report type" ;
    byte object(report) ;
        object:long_name = "echo object" ;
        object:units = "enumerated in rarep.h" ;
    byte ncover(report) ;
```

```

        ncover:long_name = "number of coverage-groups" ;
byte coverage(report, maxcover) ;
        coverage:long_name = "amount of coverage" ;
        coverage:units = "1/10" ;
byte nprecip(report, maxcover) ;
        nprecip:long_name = "number of precipitation-types" ;
byte precip_type(report, maxcover, maxprecip) ;
        precip_type:long_name = "precipitation type" ;
        precip_type:units = "enumerated in rarep.h" ;
byte intensity(report, maxcover, maxprecip) ;
        intensity:long_name = "echo intensity" ;
        intensity:units = "enumerated in rarep.h" ;
byte trend(report) ;
        trend:long_name = "intensity trend" ;
        trend:units = "enumerated in rarep.h" ;
byte nazran(report) ;
        nazran:long_name = "number of AZRANs" ;
byte nwidth(report) ;
        nwidth:long_name = "number of widths" ;
byte nmove(report) ;
        nmove:long_name = "number of movements" ;
byte nheight(report) ;
        nheight:long_name = "number of heights" ;
float azimuth(report, maxazran) ;
        azimuth:long_name = "echo-object azimuth" ;
        azimuth:units = "degrees" ;
float range(report, maxazran) ;
        range:long_name = "echo-object range" ;
        range:units = "kilometers" ;
float height(report, maxheight) ;
        height:long_name = "echo-object altitude" ;
        height:units = "meters" ;
float width(report, maxwidth) ;
        width:long_name = "echo-object width" ;
        width:units = "kilometers" ;
float move_speed(report, maxmove) ;
        move_speed:long_name = "echo-object movement-speed" ;
        move_speed:units = "m/s" ;
float move_dir(report, maxmove) ;
        move_dir:long_name = "echo-object movement-direction" ;
        move_dir:units = "degrees" ;

// global attributes:
        :title = "radar observations" ;
        :version = "2.0" ;
        :history = "Radar converted file from WXP decoders" ;
}

```

FILES

- **rad.cty** -- used to located radar sites on the LFM grid

SEE ALSO

- [radcvt](#)
- [rad](#)

ASCII RCM Radar File

This type of file is generated by the **radcv**t program for use with the WXP radar display program.

FORMAT

The format of the file is as follows:

```

WXPRAD
hhnnZ dd mmm yy
RCM SUMMARY
+ rr
echos
+ rr
...
** id num mode
T ss lat lon data
T ss lat lon data
...
** id num mode
...

```

Header Format

The first line of the file is the string:

```

WXPRAD

```

which is used to determine file type. The second line of the data file contains the date and time in the following format:

```

hhnnZ dd mmm yy

```

Where:

- *hhnn* -- Hour and minute of the observation in GMT
- *dd* -- Day of the observation
- *mmm* -- A three letter abbreviation for the month
- *yy* -- The last two digits of the year

Example: **21Z 14 JUN 98**

Radar Summary Data

The radar summary data are displayed on the following lines. Line three of the data file marks the start of the radar summary information:

```

RCM SUMMARY

```

This is followed by a row identifier line containing the following:

```

+ rr
echos

```

Where:

- *rr* -- The row number on a 360 row by 452 column grid.
- *echos* -- echo information. A total of 452 echo data points.

The following couple of lines contain echo data. There is a total of 452 echo points for each row spread out over 7 lines. The echo data is followed by another row identifier and another 7 lines of echo data. This is repeated until all 360 rows of the summary have been written to the file.

```
RCM SUMMARY
+ 0
...
+ 90
                                     11 1
                                     11 252 422221 11 11111 211 1
1          523          42          122 5642 1          1 1211
                                     1
+ 91
                                     22 1 1          2 1462 2541222 111111121 1522 11
1          1          21 365 1 1 1 11 11 121 1
                                     2
+ 92
                                     1
11          1          1          251 461 2351221 11 1 22111 11
                                     5553 1311114111112 13212 1
+ 93
...
```

Radar Site Data

After the radar summary is listed, decoded information from each RCM report is listed for radar annotation. Each radar site is delimited with a header line with the station ID, station number and mode. Storm information follows:

```
** id num mode
T ss lat lon data
T ss lat lon data
...
** id num mode
...
```

Where:

- *id* -- Station identifier (3 letter)
- *num* -- Station number (3 digit)
- *mode* -- Station mode
 - PCPN** -- Precipitation mode
 - CLAR** -- Clear air mode
 - NE** -- No Echo
 - NA** -- Not Available
 - OM** -- Out for Maintenance
- *T* -- Type of storm report
 - Z** -- Max Echo Top location
 - T** -- TVS (Tornado Vortex Signature) location
 - M** -- MESO (Mesocyclone) location
 - S** -- Storm location
- *ss* -- Storm ID
 - If type is **Z**, *ss* is the max echo top in 100s of feet
- *lat lon* -- Location of storm in latitude and longitude

- *data* -- Storm data
 If type is **S**, *data* is "*dir spd top hail*", where "*dir spd*" is the direction (in degrees) and speed (in knots) of the storm, "*top*" is the echo tops of the storm (in 100s of feet) and *hail* is the hail probability (0=none, 1=possible, 2=likely).

An example of converted RCM reports are listed below:

```

** BMX 320 CLAR
Z 30 33.461 -86.498
** EOX 362 CLAR
Z 280 30.750 -83.470
** HTX 826 CLAR
Z 70 34.990 -86.217
** MOB 509 PCPN
Z 320 28.647 -88.583
S 00 29.715 -88.939 056 006 151 0
** MXX 354 CLAR
Z 130 32.641 -87.593
** LZK 395 PCPN
Z 530 35.064 -92.716
S A1 34.592 -93.176 287 003 398 1
S H0 34.799 -92.786 133 008 315 1
S H2 34.614 -92.276 200 008 282 1
S P7 34.956 -92.632 072 004 490 0
S N0 34.540 -93.968 217 013 361 0
S L1 35.169 -91.782 315 007 323 0
S W2 34.452 -93.989 293 005 207 0
S D1 34.937 -92.525 355 054 282 0
S U2 35.081 -91.807 278 011 284 0
S R1 34.717 -93.926 287 006 193 0
S F2 35.006 -92.394 296 034 210 0
S E8 34.986 -92.286 341 011 150 0
    
```

Describing the information from LZK (Little Rock AR). The site number is 395 and it is in PCPN (precipitation) mode. The max top is 53,000 feet and 35.064N, 92.716W. The first storm is "A1" which is at 34.592N, 93.176W. Its movement is 287 degrees at 3 knots. The max echo top is 39,800 feet and hail is possible.

EXAMPLES

A sample radar converted file would look like:

```

WXPRAD
1915Z 3 AUG 98
RCM SUMMARY
+ 0
...
+165

      1111
1221111 11 1 221242221111 111562 343 1 1 1 1 11 11 122
      34423221 11 1111
      1 1 11 1211111111 1
+166

      111
12211112111211311143221122 1354 12 1 1 1 111 1221
      32222451 221 111 1
      11 11 1111 1 1
    
```

```

      21  11      121121111
+167
          1
      532211113441  1 133331121      1133
          1  413661145421 11111
          1          1      1111
      1  21  1111      112      1
+168
          1111
      234221562      11      11
          1343353116631221111
          1          1
      1  11  1111      11 11  11
+169
...
** BMX 320 CLAR
Z  30  33.461 -86.498
** EOX 362 CLAR
Z  280  30.750 -83.470
** HTX 826 CLAR
Z  70  34.990 -86.217
** MOB 509 PCPN
Z  320  28.647 -88.583
S  00  29.715 -88.939 056 006 151 0
** MXX 354 CLAR
Z  130  32.641 -87.593
** LZK 395 PCPN
Z  530  35.064 -92.716
S  A1  34.592 -93.176 287 003 398 1
S  H0  34.799 -92.786 133 008 315 1
S  H2  34.614 -92.276 200 008 282 1
S  P7  34.956 -92.632 072 004 490 0
S  N0  34.540 -93.968 217 013 361 0
S  L1  35.169 -91.782 315 007 323 0
S  W2  34.452 -93.989 293 005 207 0
S  D1  34.937 -92.525 355 054 282 0
S  U2  35.081 -91.807 278 011 284 0
S  R1  34.717 -93.926 287 006 193 0
S  F2  35.006 -92.394 296 034 210 0
S  E8  34.986 -92.286 341 011 150 0

```

FILES

- **rad.cty** -- used to located radar sites on the LFM grid

SEE ALSO

- **radcvf**
- **rad**

NIDS Radar File

NIDS is the format for disseminating NEXRAD radar data. The format compresses the data for easier transmission. WXP handles the major NIDS data types that are broadcast as part of the NIDS data service including radial and raster images, VAD winds and the storm attribute tables listed in the composite reflectivity product. The **rad** program will decode and display NIDS files.

SEE ALSO

- [rad](#)

NOWRad Radar File

NOWRad is the WSI radar mosaic product. There is a 2km and 8km resolution reflectivity product which can be displayed by the **rad** program.

SEE ALSO

- [rad](#)

Albany NLDN File

NLDN (National Lightning Detection Network) data reports lightning strike data for the contiguous U.S. The Albany NLDN format is available to the Unidata sites as part of the LDM/IDD. The file contains location (latitude, longitude), intensity and stroke count for each lightning strike. The **light** program will decode and display NLDN files.

SEE ALSO

- [light](#)

Unisys NLDN File

NLDN (National Lightning Detection Network) data reports lightning strike data for the contiguous U.S. The Unisys NLDN format is available from the Unisys WeatherMax system. The file contains location (latitude, longitude) and intensity (+ or - only) for each lightning strike. The **light** program will decode and display NLDN files.

SEE ALSO

- [light](#)

ASCII/Binary Grid File

This type of file is generated by the **sfcalc**, **upcalc**, **focalc**, **grbcalc**, **griblook**, **grid** and **grdmath** programs for use with other WXP grid display programs such as **contour** and **vector**.

FORMAT

The contents of the gridded files are standardized to work with all WXP programs. They contain the following information:

WXPGRID

```
date
[flag-]info
[src ver model grid ltype level ftime var:units]
proj clat clon dx dy nx ny
grid...
```

Header Data

The first line of the file is the string:

```
WXPGRID
```

which is used to determine file type. The second line of the data file is a character string describing the valid time and origin of the image (maximum 50 characters). Examples:

```
12Z 14 JUN 98
WXP analysis for 12Z 14 JUN 98
24 hour NGM valid 00Z 2 JUN 98
```

The third line contains a character string describing the contents of the image (maximum 50 characters). Examples:

```
Integrated Cloud Coverage (%)
IR Temperature (C)
```

The third line may contain a flag which describes file format. The flag can be:

- **2** -- grid file with extended header and ASCII gridpoint data
- **3** -- grid file with extended header and binary gridpoint data.

Extended Header Data

The grid description line contains grid description variables in the following format:

```
src ver model grid ltype level ftime var:units
```

Where:

- *src* -- Grid source,
- *ver* -- Grid version,
- *model* -- Grid model,
- *grid* -- Grid type,
- *ltype* -- Level type,
- *level* -- Vertical level (units depend on *ttt*),
- *ftime* -- Forecast time in hours,
- *var* -- Variable type
- *units* -- Variable's units (optional)

Example: **7 1 39 50 1 0 24 350**

The values of these variables are described in [WXP Product Description Appendix](#).

Domain Header Data

The fourth line contains image size and location in the following format:

```
proj[:param...] clat clon dx dy nx ny
```

Where:

- *proj* -- The projection:
LL -- lat-lon,
PS -- polar stereographic,
ME -- mercator,
LC -- lambert conformal. (see **plot_domain** section of Users Guide for more projections)
- *param* -- The projection parameters (see **plot_domain** section of Users Guide).
- *clat* -- The central latitude (N) of the grid.
- *clon* -- The central longitude (E) of the grid.
- *dx* -- The X coordinate grid spacing based on projection.
- *dy* -- The Y coordinate grid spacing based on projection.
- *nx* -- The number of gridpoints in the X direction.
- *ny* -- The number of gridpoints in the Y direction.

Example: **PS 39.0 -97.0 2.3 2.3 25 17**

An example of the header:

```
WXPGRID
WXP analysis for 15Z 6 AUG 98
2-Surface Temperature (F)
1 1 1 2 1 -9999 0 11:F
PS 43.00 -93.00 1.10 1.10 25 17

WXPGRID
24 hour Eta valid 12Z FRI 7 AUG 98
2-850 mb Temperature (C)
7 1 89 211 100 850 24 11:C
LC:90.00:-95.00:25.00:25.00:1.00 40.606583 -100.553223 0.812710 0.812710 93 65
```

Describing this example, the first three lines are self-explanatory. The grid description line contains the following information:

```
Grid source=7 -- National Meteorological Center (NMC)
Grid version=1 -- Version number 1.
Grid model=89 -- Eta model
Grid type=211 -- AWIPS grid 211
Level type=100 -- Pressure level
Vertical level=850 -- 850 mb
Forecast time=24 -- 24 hour forecast
Variable type=11:C -- Temperature in C
```

This is used by WXP to quickly determine grid type, variable and units.

The grid is a 93x65 Lambert conformal grid centered at 40.6 North and 100.55 West with a standard parallel of 25 North. The grid spacing is roughly 81 km.

Grid Data

Each successive line contains the gridded data in a real number format for the *xx* by *yy* grid. The data are placed in the file starting at the upper lefthand gridpoint, and the first *xx* pieces of data in the file contain the first row of data running from left west to east. Subsequent *xx* pieces of data are used for the next rows of the data file until *yy* rows are filled. If the output is ASCII, the output is using a generic format in scientific notation (%e). If the output is binary (flag=3), the output is IEEE big endian floating point (4 byte).

EXAMPLES

```
WXPGRID
24 hour Eta valid 12Z FRI  7 AUG 98
2-850 mb Temperature (C)
7 1 89 211 100 850 24 11:C
LC:90.00:-95.00:25.00:25.00:1.00      40.606583      -100.553223      0.812710
0.812710 93 65
-1.39984131e-001
-1.39984131e-001
1.10015869e-001
3.60015869e-001
6.10015869e-001
3.60015869e-001
6.10015869e-001
1.11001587e+000
...
```

SEE ALSO

- [sfccalc](#)
- [upcalc](#)
- [focalc](#)
- [grbcalc](#)
- [griblook](#)
- [grid](#)
- [grdmath](#)
- [vector](#)
- [contour](#)

GRIB Grid File

GRIB is a WMO standard for grid transmission. The gridpoint data is compressed to enable easier transmission. WXP handles most GRIB formats including all GRIB products from FOS, NOAAPORT and the NCEP NIC server.

FORMAT

The GRIB format is laid out into six sections:

1. Indicator block: the characters "GRIB" plus product size.
2. Product definition block: this block describes all data contained in the product
3. Grid definition block: this block describes in detail the layout of the grid (OPTIONAL)
4. Bit map block: this block is used to define where valid data is located is a sparse grid (OPTIONAL)
5. Binary data block: binary packed gridpoint data
6. End of record block: the characters "7777"

WXP can be used as a GRIB decoder by using either the **griblook** or **grbcalc** programs and saving the grid to a WXP ASCII grid file. Programs like **grbcalc** and **contour** can be used to view the grids.

SEE ALSO

- [griblook](#)
- [grbcalc](#)
- [contour](#)

ASCII Raw File

The raw data file contains the same information as the WXP format. This type of file is provided to allow non-WXP programs to create data that can be used by WXP. Raw data files are written in ASCII and contain location and values for various data points. This data can be plotted or fit to a grid for contouring. Raw data files can also be used to plot and annotate various plots with text and weather symbols. These files can be created by many of the plotting programs such as **sfcwx**, **upairwx**, and **fouswx**. Raw data can be displayed using **mapplt**, fitted to a grid with **grid**, and manipulated with **rawmath**.

FORMAT

The format is as follows:

```

WXPRAW
date
[flag]info
[data header]
raw data...

```

Header Data

The first line of the file is the string:

```
WXPRAW
```

which is used to determine file type. The second line of the data file is a character string describing the valid time and origin of the image (less than 50 characters). Examples:

```

12Z 14 JUN 98
May 1995

```

The third line contains a character string describing the contents of the raw file (less than 50 characters). Examples:

```

Surface Temperature (C)
Monthly Precipitation (in)

```

Data Header

There is an optional data header which is placed after the information line which describes the raw data. This is not necessary if the only one type of data exists in the file but it is often more convenient to put more than one piece of data for each station in the raw file. This is done by specifying more than one value on each line of station data. In order to access this data, the column number can be specified or the column name. The name and additional attributes are specified in the data header.

To include the data header, a "!" (exclamation point) flag must be added to the beginning of the information line. The data header contains a list of information, whitespace delimited, one for each column of data. The format is:

```
name[:long_name][:attributes...]
```

where:

- *name* -- The name used to describe the column of data. This is used with programs like **mapplt** to extract this data from the raw file.
- *long_name* -- This is a long name that will be printed at the top of a plot using this data.
- *attributes...* -- This is a set of plotting attributes (see [data plotting attributes](#) in the Users Guide). This can also include plotting types such as **wx**, **mark**, **wbarb**, etc.

The data header block is terminated with a "\$". The rest of the line following the "\$" is ignored. This can be used for column headers. Here is an example:

```

WXPRAW
22-30 JUN 1998
!Hurricane BLAS
ID LAT LON
TIME:Time:ur:tx
WIND:Max_wind_(kt):ul:tx
PR:Low_central_pressure_(mb):ll:tx
STAT:Storm_status:co=yellow
LINE:Storm_track
$ID  LAT  LON  TIME  WIND  PR  STAT  LINE
   1  9.90 -96.20 22/15Z  30   -   L  post:line:co=green  TROPICAL_DEPRESSION

```

Raw Data

The following lines contain information for each station/report. The format is:

```
ident [lat lon] data1 [data2...]
```

Where:

- *ident* -- The unique station/report identifier (up to 10 characters)
- *lat lon* -- Latitude, longitude location data. For some types of raw files, this will be *X Y* position values or a single pressure value *P*. These fields can be omitted if only one data value *data1* appears after the *ident* or if a dash '-' appears after *ident*.
- *data1* -- This is a single data value. The syntax is:

```
[type:]data[:attrib...]
```

where:

- *type* -- Data type identifier. This identifies the type of data. This is optional if there is only one type of data in the file or the column has been identified in the data header block (see above). This can be used by **mapplt** to select specific pieces of data from the raw file.
- *data* -- The actual data value. This can be a number or string.
- *attrib...* -- A set of plotting attributes (see [data plotting attributes](#) in the Users Guide). These attributes will override those listed in the data header. This can also include plotting types such as **wx**, **mark**, **wbarb**, **line**, etc.
- *data2* Subsequent data values for a particular station/report. These are separated by a space and follow the same format as *data1*. These data can be accessed by specifying the column number or name (as described below).
-

Raw files should have one blank line at the end. This way raw files can be strung together in a stream to programs such as **rawmath**.

Comments

Any line starting with a "#", is considered a comment line.

EXAMPLES

A simple raw file would look like:

```

WXPRAW
19Z 21 JAN 98
Surface Temperature (F)
KBIL 45.80 -108.53 47.0
KBNA 36.12 -86.68 26.0
KCVG 39.05 -84.67 16.0
KDAY 39.90 -84.20 13.0
KDBQ 42.40 -90.70 18.0
(blank line)

```

The data lines contain a station identifier, latitude, longitude, and a simple value. Using multiple columns in a raw file would look like:

```

WXPRAW
15Z 2 NOV 97
Surface data
#ID  LAT   LON   T    TD  WIND ALT SLP VIS
KAUG 44.32 -69.80 34.0 20.0 0207 040 399 15
KBGR 44.80 -68.82 35.0 19.0 0206 040 297 15
KBNA 36.12 -86.68 62.0 49.0 2311 970 054 10
KCAR 46.87 -68.02 30.0 18.0 3509 037 297 30
(blank line)

```

Here, the columns are unnamed but there is a column header as a comment line. Plotting programs will plot all of the data fields for a particular station/report. This will overlap information. To select a particular column of information, specify the column number starting with column one (in this case one is T). For example:

```
mapplt sfc.raw-1
```

To eliminate overlap, attributes may also be added to enhance output:

```

WXPRAW
21Z 21 JAN 98
Surface Station data
KABR 45.45 -98.43 C:cldcv:co=white 19:text:ul:co=green 12:text:l1:co=green
KBBW 41.43 -99.65 C:cldcv:co=white 53:text:ul:co=green 26:text:l1:co=green
KBIL 45.80 -108.53 O:cldcv:co=white 49:text:ul:co=green 24:text:l1:co=green
KBNA 36.12 -86.68 C:cldcv:co=white 28:text:ul:co=green 16:text:l1:co=green
KDLH 46.83 -92.18 S:cldcv:co=white 23:text:ul:co=green 11:text:l1:co=green

```

To simplify this, attributes can be in the data header:

```

WXPRAW
06Z 8 AUG 98
!Surface All data
id lat lon
temp:Surface_Temperature_(F):value:ul
dewp:Surface_Dewpoint_temperature_(F):value:l1
pres:Sea_level_Pressure_(mb):data:ur
wbrbc:Surface_:cbarb:
cldcv:Surface_Cloud_cover:cloud:
wx:Surface_Present_weather:symb:cl
$
KJDN 47.33 -106.93 75.000000 51.099998 097 300.000000,14.000000 - -
KPIT 40.50 -80.22 73.900002 61.000000 235 120.000000,7.000000 C -
KCID 41.88 -91.72 66.900002 66.900002 152 0.000000,0.000000 O -
KLIT 34.73 -92.23 77.000000 75.199997 176 150.000000,3.000000 B -
KIAD 38.95 -77.45 73.900002 71.099998 251 0.000000,5.000000 S -

```

```

KROW  33.30 -104.53 77.000000 50.000000 113 160.000000,8.000000 C -
KBKW  37.78 -81.12 69.099998 63.000000 220 120.000000,6.000000 C -
KRDU  35.87 -78.78 75.900002 73.900002 227 80.000000,5.000000 O TR-

```

Once the data header is specified, individual columns can be plotted using:

```
mapplt sfc.raw-temp
```

The output will be a plot of temperatures with the label of "**Surface All data-Surface Temperature (F)**" which is the combination of the raw file information line and the long name for the specified variable.

The data header is nice for specifying the names of regular columns but this doesn't work well if the data is irregular as is the case with SHEF data. In this case, the data type is specified with each data value:

```

WXPRAW
 0Z 28 DEC 93
SHEF Data
HLTK1 TA:17 TX:36 TN:17
EART2 PCIRG:16.59 PPKRG:.00
KNBT2 PCIRG:12.00 PPKRG:.00 HGIRG:2.85
NLKT2 PCIRG:9.74 PPKRG:.00
TKLT2 PCIRG:10.33 PPKRG:.00 HGIRG:4.23
BILT2 PCIRG:7.81 PPKRG:.00
STST2 PCIRG:12.93 PPKRG:.00
BCPT2 PCIRG:12.41 PPKRG:.00
ATKM5 HGIRG:7.60
RKFM5 HGIRG:5.30
ANKM5 HGIRG:7.64
SSPM5 HGIRG:87.01

```

To plot the PCIRG data, use:

```
mapplt shef.raw-PCIRG
```

A more complete example is a hurricane track raw file which includes various types of formatting:

```

WXPRAW
22-30 JUN 1998
!Hurricane BLAS
ID LAT LON
TIME:Time:ur:text
WIND:Max_wind_(kt):ul:text
PR:Low_central_pressure_(mb):ll:text
STAT:Storm_status:co=yellow
LINE:Storm_track
$ID  LAT  LON  TIME  WIND  PR  STAT  LINE
1    9.90 -96.20 22/15Z  30    -    L    post:line:co=green
TROPICAL_DEPRESSION
2    10.90 -97.90 22/21Z  45  1000 @:wx post:line:co=yellow TROPICAL_STORM
3    11.20 -98.10 23/03Z  55  997  @:wx post:line:co=yellow TROPICAL_STORM
4    11.20 -99.00 23/09Z  55  994  @:wx post:line:co=yellow TROPICAL_STORM
5    12.00 -100.30 23/15Z  55  994  @:wx post:line:co=yellow TROPICAL_STORM
6    12.80 -101.80 23/21Z  65  987  @+:wx post:line:co=red    HURRICANE-1
7    13.10 -103.40 24/03Z  90  970  @+:wx post:line:co=lred   HURRICANE-2
8    13.40 -104.10 24/09Z  95  970  @+:wx post:line:co=lred   HURRICANE-2
9    14.00 -104.70 24/15Z  100 965  @+:wx post:line:co=magenta HURRICANE-3
10   14.70 -105.70 24/21Z  100 965  @+:wx post:line:co=magenta HURRICANE-3
11   15.00 -106.50 25/03Z  115 948  @+:wx post:line:co=lmagenta HURRICANE-4
12   15.50 -107.30 25/09Z  120 945  @+:wx post:line:co=lmagenta HURRICANE-4

```

```

13 16.10 -108.10 25/15Z 120 948 @+:wx post:line:co=lmagenta HURRICANE-4
14 16.60 -108.90 25/21Z 115 948 @+:wx post:line:co=lmagenta HURRICANE-4
15 17.10 -110.10 26/03Z 100 960 @+:wx post:line:co=magenta HURRICANE-3
16 17.30 -111.00 26/09Z 95 965 @+:wx post:line:co=lred HURRICANE-2
17 17.40 -112.40 26/15Z 100 960 @+:wx post:line:co=magenta HURRICANE-3
18 17.60 -113.50 26/21Z 90 970 @+:wx post:line:co=lred HURRICANE-2
19 17.70 -114.40 27/03Z 90 970 @+:wx post:line:co=lred HURRICANE-2
20 17.80 -115.60 27/09Z 80 977 @+:wx post:line:co=red HURRICANE-1

```

The identifiers are the advisory numbers. The columns are specified in the column header section:

Col	Name	Longname	Attribute
1	ID		
2	LAT		
3	LON		
4	TIME	Time	ur:text (place upper right, data is text)
5	WIND	Max_wind_(kt)	ul:text (place upper left, data is text)
6	PR	Low_central_pressure_(mb)	ll:text (place lower left, data is text)
7	STAT	Storm_status	col:yellow (color yellow)
8	LINE	Storm_track	

Additional attributes can be specified with the data as is the case with the storm status. The "@:wx" specifies to plot a tropical storm symbol. The storm track data has a plot type of "**line**" and a data value of "**post**" which means a line will be drawn between the current location and the next location. The data value of "**pre**" can be used to draw to the previous location. The data header section is terminated with the '\$' at the beginning of the line with the header remarks. This shows using the remainder of this line as a column header line. This is considered a comment and does not affect the interpretation of the raw file. Again, a single data type can be selected by specifying the column name. For example:

```
mapplt track.dat-LINE:wi=2
```

will plot the storm track with a width of 2 pixels and changing colors based on storm intensity. The plot output will be labeled "**Hurricane Blas-Storm track**".

SEE ALSO

- [sfcwx](#)
- [upairwx](#)
- [fouswx](#)
- [mapplt](#)
- [grid](#)
- [rawmath](#)

McIDAS AREA File

This is an image file format used by the McIDAS analysis package. WXP supports a subset of the AREA file types including those with GVAR, GOES, MSAT, PS and MOLL navigation types. WXP can handle both the 8 and 16 bit image file format. Also, WXP can read in the run length compressed files provided from SSEC as part of the Unidata McIDAS data channel.

FORMAT

The file is split up into 3 components:

1. Image Directory (256 bytes) which contains the image type, size, resolution and date information.
2. Navigation (512+ bytes) which contains information to navigate the images which allows a program to convert the location of each pixel in the image to a latitude and longitude for precise location of the image.
3. Pixel data (the rest of the file). This may be packed or unpacked and its size is defined by the image directory. Each pixel/byte is saved from left to right on the scan line. Each scan line is stored consecutively from top to bottom.

SEE ALSO

- [xsat](#)

NOAAPORT Satellite File

This is an image file format used by NOAAPORT for transmission of satellite images. WXP supports the GOES NOAAPORT image format which is based on the GRIB format. These are 8 bit image files with a product definition block (similar in format to the GRIB PDB) which describes the origin and type of image as well as the projection the image is on in order to navigate the image. All NOAAPORT images are either in polar stereographic, mercator or lambert conformal projections and cover the contiguous US (CONUS), Alaska, Hawaii, Puerto Rico with a composite CONUS and northern hemisphere sector.

FORMAT

The file is split up into 2 components:

4. Product definition block which contains the image type, size, resolution, date and projection information.
5. Pixel data (the rest of the file). Each pixel/byte is saved from left to right on the scan line. Each scan line is stored consecutively from top to bottom.

SEE ALSO

- [xsat](#)

Unisys Satellite File

Unisys delivers satellite imagery in this format. Most Unisys imagery is un-navigated except for some lambert conformal images. The **xsat** program can display the navigated images with a map overlay or the un-navigated images. Pseudo-navigation can be added to these images with the **grid_domain** resource.

FORMAT

The file is split up into 2 components:

1. Product header which contains simple navigation and product time. The header is ASCII data with the terminating "**end**" string. The contents of the header include keyworded pieces of data. The keywords are:
 - **hsxxx**-- Header size in bytes (not reliable)
 - **xxxx** -- Image width in pixels
 - **yxxx** -- Image height in pixels
 - **wrxxx** -- Image width resolution
 - **hrxxx** -- Image height resolution
 - **idxxx** -- Sector name
 - **bxxx** -- Sensor band number
 - **ssnxxx** -- Start element number on scan line
 - **slnxxx** -- Start scan line number
 - **esnxxx** -- End element number on scan line
 - **elnxxx** -- End scan line number
 - **srxxx** -- Spot/element resolution
 - **lrxxx** -- Line resolution
 - **bppxxx** -- Bits per pixel
 - **latxxx** -- Central latitude
 - **lonxxx** -- Central longitude
 - **ref1xxx** -- Reference latitude
 - **ref2xxx** -- Reference longitude
 - **ref3xxx** -- True latitude #1
 - **ref4xxx** -- True latitude #2
 - **fldxxx** -- Date of image
 - **end** -- End of header
2. Pixel data (the rest of the file).

EXAMPLES

Here is a lambert conformal image over the US:

```
hs223  x640 y448 idnav_ir b1 fld902803530
bpp8 pl piset0 gg
lat38.000000 lon-98.000000 range31.540000
ref138.000000 ref2-98.000000 ref333.000000 ref445.000000
maxrank255 CIL2 RIV2 PBY1 BDY1 res255
dlat0.00000000 dlon0.00000000      end
~~K5459;EUOD<9<@=@=<;?<47<=9:>96544444444444444444665566676
...
```

Here is a un-navigated GOES image:

```
hs109  x1024 y1024 wr25000 hr25000 idful_di b1 ssn1420 sln324
esn6241 eln1715 sr4 lr4 fld902805362 v10137 end
~~~~~
...
```

SEE ALSO

- [xsat](#)

WXP Image File

This type of file is a generic image format that can be displayed with the WXP **xsat** satellite display program. This can be used to import image data into WXP.

FORMAT

The format of the file is as follows:

```

WXPIMG
date
info
proj clat clon dx dy nx ny
image...

```

Header Data

The first line of the file is the string:

```

WXPIMG

```

which is used to determine file type. The second line of the data file is a character string describing the valid time and origin of the image (less than 50 characters). Examples:

```

1215Z 4 AUG 98
ISCCP analysis valid 00Z 2 JUN 90

```

The third line contains a character string describing the contents of the image (less than 50 characters). Examples:

```

GOES East IR 11um
Integrated Cloud Coverage (%)

```

The fourth line contains image size and location in the following format:

```

proj[:param] clat clon dx dy nx ny
or
nx ny

```

Where:

- *proj* -- The projection:
 - LL** -- lat-lon,
 - PS** -- polar stereographic,
 - ME** -- mercator,
 - LC** -- lambert conformal,
 - SAT** -- satellite projection. (see **plot_domain** section of Users Guide for more projections)
- *param* -- The projection parameters (see **plot_domain** section of Users Guide).
- *clat* -- The central latitude (N) of the grid.
- *clon* -- The central longitude (E) of the grid.
- *dx* -- The X coordinate pixel spacing based on projection.
- *dy* -- The Y coordinate pixel spacing based on projection.
- *nx* -- The number of pixels in the X direction.
- *ny* -- The number of pixels in the Y direction.

Examples:

```

ME 30.0 -100.0 0.187793 0.187793 640 427

```

LL 0.0 0.0 2.5 2.5 144 72
1000 760 (if not geographic data, just specify size)

An example of the five line header is listed below:

```
WXPIMG
0615Z  9 AUG 98
GOES NH Infrared 1lum
ME 30.000000 -100.000000 0.187793 0.187793 640 427
```

```
WXPIMG
ISCCP analysis valid 00Z 2 JUN 90
Integrated Cloud Coverage (%)
LL 0.0 0.0 2.5 2.5 144 72
```

Image Data

The image follows a newline character terminating the projection line. The image is binary, stored 8 bits per pixel which gives an intensity range of 0 to 255. The size of the image is defined by **ny**, number of scan lines, and **nx**, number of elements per scan line. Therefore, there are **nx** bytes per scan line. Each pixel/byte is saved from left to right on the scan line. Each scan line is stored consecutively from top to bottom.

EXAMPLES

A sample image file would look like:

```
WXPIMG
0615Z  9 AUG 98
GOES NH Infrared 1lum
ME 30.000000 -100.000000 0.187793 0.187793 640 427
vssqoknux{~~<81><92><A1><AC><AB><AC><AE><A6><A1><9F><9F><A2><A8><A3><9A><93><92>
<92><8E><8E><93><97><9E><A7><AB><A5><9E><A4><AC><8E>uii jk jhi jlnpsrqur ihgghi jmpq
rtutps~<90><A1><A9><A4><9D><96><99><A1><AB><AF><AE><AF><B2><B5><AD><A8><A8><A8>
```

The image file can be displayed using:

```
xsat -inp=img file.img
```

SEE ALSO

- [xsat](#)

Bulletin File (.bul)

The ingest programs uses a bulletin file to set up which products are to be selected from the data feed and which actions to perform on them. The bulletin filename is specified with the [bull_file](#) resource.

FORMAT

The bulletin file contains a list of headers, actions and commands to be performed:

```
header [action] [command/filename...] [header file]
header [action] [command/filename...] [header file]
...
```

The header can specify the exact header or a pattern to which headers can be matched. The headers listed in the file can use the following wildcard characters:

. or ?	match a single character
- or *	match any character
[letters]	match a character from the set.
[^letters]	match any character except those from the set
(str1 str2...)	match strings
_	underscore matches a space
/data	match extra information

Some example header strings are:

AB	Anything that starts with AB
S[AP]	SA or SP
(W AC RG)	Starts with W or AC or RG
F[^O]	Anything that starts with F, second character NOT O
FQUS1_KIND	Full header specification with spaces as underscores
*_KIND	Wildcard match on any product that ends with KIND

When the product is GRIB, the header is parsed for specific product parameters. This information can then be used to select the product. The syntax for this selection is:

```
/[Xvvv][Xvvv][Xvvv]...
```

Where X is:

- M** -- model number
- G** -- grid number
- L** -- level type
- H** -- level value
- T** -- forecast time
- V** -- variable number
- vvv -- the value of the parameter

The values for each parameter are listed in the [WXP Product Description Appendix](#). Using the internal GRIB parameters is more reliable than selecting by the WMO header because more than one product may have the same header:

```
HVAC98 KWBC 070000 from Sea Wave model
HVAC99 KWBC 070000 from Aviation model
```

To separate the two products, use the model specifications: **/M77** for the Aviation model and **/M10** for the Sea Wave model.

Actions

The actions are:

>>	append to file with header
append	same as above
>	write to file with header, previous content overwritten
write	same as above
#	write to file without header, previous contents overwritten
file	same as above
	pipe product to listed command
pipe	same as above
@	run command when product complete
run	same as above

Also, the action can be prepended by a set of flags:

- **R** -- specifies to save the file as a raw file and not strip control characters.
- **B** -- specifies a product to be a binary product and not strip unprintable characters
- **P** -- specifies to send a PAN message at the completion of a product

Command or Filename

The command is generally the file to place the output or the command to run with the pipe or run actions. The command can have several escape characters:

**Examples based on system time 1455Z Jan 12, 1997,
product header FPUS5 KIND 281512**

Wildcard	Explanation	Example
@tag	Name convention tag	
%Y	current system year	1997
%y	current system year (last 2 digits)	97
%m	current system month	01
%d	current system day	12
%j	current system Julian day	12
%h	current system hour	14
%n	current system minute	55
%pd	product day	28
%ph	product hour	15
%pn	product minute	12
%T	product type	FPUS5
%t	product type (lower case)	fpus5
%L	product locale	KIND
%l	product locale (lower case)	kind
%D	data_path resource	
%C	con_path resource	
%R	raw_path resource	
%G	grid_path resource	
%W	watch_path resource	
%I	image_path resource	

%F	file_path resource	
-----------	--------------------	--

Some of the above wildcards can be preceded with a number. For dates, the number is a modifier which rounds down to the nearest value which is a multiple of that number. For example, "**%6h**" would round down to the nearest 6 hour boundary. For the previous example, it results in the value 12.

For the product type and locale, this number is used in a substring operation. The first digit of the number is the offset into the string and the second digit refers to the number of characters to use. For example, "**%12T**" results in "**FP**". To get "**IND**", use "**%23L**".

Header Files

To aid in the parsing of products from the various feeds, a header file can be created by the ingest program. This essentially lists the header of each product in the file along with its byte offset into the file. Since most parsing is based on header, it is far easier to search the smaller header file than to parse through the much larger product file. To produce these files automatically by the ingestor, add the file name convention to the end of the line in the bulletin file:

```
F[^O]          >> %D/%y%m%d%6h_for.wmo %D/%y%m%d%6h_for.hdr
```

The first name convention listed "**%D/%y%m%d%6h_for.wmo**" is the filename where the actual product is saved. The second name convention "**%D/%y%m%d%6h_for.hdr**" is where the header file information is saved. The syntax of the file is as follows:

```
offset header / extra
offset header / extra
....
```

where:

- offset -- is the byte offset into the file,
- header -- is the product header in its entirety is listed after the offset
- extra -- extra information about the product which is normally the AWIPS header

A sample from a forecast data header file:

```
0 FPUS86 KPQR 282359 / OPUPDX
3264 FPUS85 KGGW 290001 / OPUGGW
3548 FPAK11 PAYA 282207 / &ZCZC JNULFPYAK
4190 FPUS73 KFGF 282359 / NOWFAR
```

For more information on header files, see the section on header files.

PAN (Product Arrival Notices) Messages

To set up the WXP ingestor for PAN messages the following pieces of information must be added to the bulletin file. At some point in the file, a PAN configuration line must be added.

```
# PAN Setup
@PAN id=45 sock:steve:5566 sock:dev5:5000 pan.log
```

The "**@PAN**" is a keyword in the bulletin file for the PAN configuration line. The "**id=45**" specifies the NOAAPORT unique server ID which is broadcast as field 2 in the PAN message. The rest of the line lists destinations. The "**sock**" keyword specifies the PAN go over a UDP socket. The string "**steve:5566**" is the network name of the destination computer and the TCP/IP port number. If the sock keyword is omitted, the PAN is save to the listed filename such as "**pan.log**". Up to 10 destinations can be listed. Each destination is addressed starting with 0 and going to 9 in the order listed on the PAN line.

By default, no PAN messages are sent even if the PAN line is added to the bulletin file. To enable PAN messages, the "P" flag must be added to the action for each product being saved on the server. For example a product line would look like:

```
# Pattern Action Filename Header Filename
FT      >> %D/%y%m%d%h_term.wmo %D/%y%m%d%h_term.hdr
```

To enable this product type for PAN messages, add the "P" flag to the action.

```
FT      P>> %D/%y%m%d%h_term.wmo %D/%y%m%d%h_term.hdr
```

This will send a PAN message to all listed destinations whenever this products is received. If you don't want to send a PAN to all destinations, the destination IDs can be listed:

```
FT      P035>> %D/%y%m%d%h_term.wmo %D/%y%m%d%h_term.hdr
```

In this case, PAN messages will only be sent to the 0, 3 and 5th destinations.

EXAMPLES

```
# Pattern          Action Filename          Header Filename
#
S[AP]              >>-15 %D/%y%m%d%h_sao.wmo
S[IMNS]            >>-05 %D/%y%m%d%h_syn.wmo
SD                 >>+07 %D/%y%m%d%h_rad.wmo
U[^AB]             >>-65 %D/%y%m%d%12h_upa.wmo
ASUS1_             >>      %D/%y%m%d%3h_frt.wmo
WWUS40             >>      %D/%y%m%d%6h_wws.wmo
FO                 >>      %D/%y%m%d%12h_mod.wmo %D/%y%m%d%12h_mod.hdr
A                  >>      %D/%y%m%d%6h_sum.wmo %D/%y%m%d%6h_sum.hdr
C                  >>      %D/%y%m%d%6h_cli.wmo %D/%y%m%d%6h_cli.hdr
W                  >>      %D/%y%m%d%6h_sev.wmo %D/%y%m%d%6h_sev.hdr
#
# Specific forecast products
#
FXUS01             >      %D/fore/48hr
FXUS02             >      %D/fore/3-5d_Hem
FPUS5_KIND         |      /usr/local/bin/parse - -ph=FPUS5_KIND -id=%INZ029 -pa=dollar -
of=%D/fore/laf_zone -me=none
*_KIND             >>      %D/Indy/%m%d.dat
#
# HDS products
#
Y/M89              >>      %D/%y%m%d%12h_eta.grb %D/%y%m%d%12h_eta.hdr
Y/M39G211          >>      %D/%y%m%d%12h_ngm.grb %D/%y%m%d%12h_ngm.hdr
Y/M64G211          >>      %D/%y%m%d%12h_ngm.grb %D/%y%m%d%12h_ngm.hdr
```

SEE ALSO

- [ingest](#)

Case Study Lookup File (case.lup)

To setup the case studies, the "**case.lup**" is used. This file lists each case study and the resources to set and is located in the WXP database directory (see [file_path](#)).

FORMAT

There are two types of lines in this file. The first is a header line. The syntax is:

header *description*

Where:

- *description* -- A description of a particular set of case studies (ie Winter Storms)

The second type is the case line. The syntax is:

case *name alias alias*
description
resource value
resource value
 ...

Where:

- *name* -- The name to use for the case. This is usually a short abbreviation for the case such as a date or special event. This also is the string to be displayed in the WXP shell when the **case** command is run. The name can be "**nolist**" which specifies to not list the case study in the case listing. This allows you to setup large numbers of cases without making the case listing in the WXP shell to large to view.
- *alias* -- A set of aliases to be used for the case.
- *description* -- A full description of the case. This can be up to 80 characters.
- *resource* -- The name of a WXP resource to set. There can be any number of resources set. This actually sets an environment variable so that command line parameters will override this setting.
- *value* -- The value of the resource.

EXAMPLES

Here is a sample:

```
header *** Winter Cyclones ***

case nolist 20nov94
Cyclone Occlusion over Plains and Midwest (20 Nov 1994)
data_path /home/wxp/case/20nov94
con_path /home/wxp/case/20nov94
curtime 199411220000

case 07jan97 444
Cyclone and Snow event over Midwest (07-10 Jan 1997)
data_path /home/wxp/case/07jan97
con_path /home/wxp/case/07jan97
curtime 199701091200
```

The header line is used to list the following case studies as winter cyclones. The first case "**20nov94**" will not be listed when the "**case**" command is invoked but still can be used. The second line "Cyclone..." is a long description of the case. The next three lines set the "**data_path**" and "**con_path**" resources with the environment variables "**wxpdata_path**" and "**wxpcon_path**". The last line sets the "**wxpcurtime**" environment variable to reset the

current time to the most important time of the case study so that the **current** resource can be used with the case study.

The WXP User's shell uses a menu to select programs and options. In order to make the shell more extensible, the menu items and the menus themselves can be changed at any time. This is done by modifying the "**wxp.menu**" file located in the WXP database directory (see [file_path](#)). This file lists all the menus, the labels used to describe each action and the command to run when the item is selected.

SEE ALSO

- [wzp](#)

City Location File (.cty)

The city database file contain reporting station location information.

FORMAT

The format for this database file is as follows:

```
longname    st cn iiii MpU lat lon elev wmo [###]
```

Where:

- *longname* -- Station name. (max 15 characters)
- *st* -- Abbreviated state or region name. (2 characters-starts in column 17)
- *cn* -- Abbreviated country name. (2 characters-starts in column 20)
- *iiii* -- ICAO station identifier. (up to 5 characters-starts column 23)
- *M* -- Indicator for forecast model output station. (Obsolete-column 29)
- *p* -- Priority used for display filtering purposes. (column 30)
1 is the highest priority, 7 is the lowest, 9 represents no priority
- *U* -- Indicator for an upper air station. (Obsolete-column 31)
- *lat* -- Latitude in degrees (north is positive)
- *lon* -- Longitude in degrees (east is positive)
- *elev* -- Elevation in meters
- *wmo* -- 5 digit WMO station identifier (**99999** represents no assigned numeric identifier). This is the site ID for radar data.
- *###* -- optional numeric field (used for station occurrence)

EXAMPLES

Here is a sample of some stations from the **sao.cty** file:

```
Phoenix/Luke AF AZ US KLUF 5 33.53 -112.38 332 99999
Philadelphia In PA US KPHL 2 39.88 -75.25 9 72408
Petersburg (AWO VA US KPTB 4 37.18 -77.52 59 99999
Petersburg AK US PAPG 4 56.82 -132.97 0 70386
Peoria Regional IL US KPIA 2 40.67 -89.68 202 72532
Pensacola Regio FL US KPNS 4 30.47 -87.18 37 99999
Pendleton Munic OR US KPDT 3 45.68 -118.85 456 72688
Pelly Bay 1 NT CN CWRP 3 69.43 -89.73 325 71918
Pellston/Emmet MI US KPLN 4 45.57 -84.80 219 99999
```

City Database Files

Database	Type of Stations	Location
sao.cty	Surface SAO/METAR	Global
sao_all.cty	Surface SAO/METAR	Global+non-reporting
upa.cty	Upper air	Global
upa_all.cty	Upper air	Global+non-reporting
syn.cty	Surface Synoptic	Global
syn_all.cty	Surface Synoptic	Global+non-reporting
rad.cty	NEXRAD Radar	United States
mos.cty	Model Output	North America
shef.cty	SHEF	United States

SEE ALSO

Color Table File (.clr)

This file contains the color table information used to set color index and names for WXP programs. In other words, it lists the valid colors that can be used in a WXP program.

FORMAT

Each line of this file contains the color information for a specific color:

```
[index] name red green blue
```

where:

- *index* -- is the color index used internally to represent the color (this is optional)
- *name* -- is an alias that can be used to represent the color index. This name is used in all color resources
- *red* -- the amount of red used in the color (floating point value from 0 to 1)
- *green* -- the amount of green used in the color (0 to 1)
- *blue* -- the amount of blue used in the color (0 to 1)

Up to 256 colors may be specified in the file. If the index is not specified, the indices start with 0 and increment by one for each line in the file.

End Keyword

A line containing just the word "**end**" represents the end of the color fill contour colors. In the first color table database, the color fill resource defaults to all the colors listed. The **end** keyword terminates the color fill sequence if the whole file is not part of the color fill pattern.

EXAMPLES

Here is the **wxp.clr** color database file.

```
Black      0.0      0.0      0.0
White     1.0      1.0      1.0
Red        0.7      0.0      0.0
Green      0.0      0.7      0.0
Blue       0.0      0.0      0.7
Yellow     1.0      1.0      0.3
Cyan       0.0      0.7      0.7
Magenta    0.7      0.0      0.7
DGray      0.3      0.3      0.3
LGray      0.7      0.7      0.7
LRed       1.0      0.3      0.3
LGreen     0.3      1.0      0.3
LBlue      0.3      0.3      1.0
Brown      0.7      0.7      0.0
LCyan      0.3      1.0      1.0
LMagenta   1.0      0.3      1.0
```

SEE ALSO

Color Fill File (.cfl)

The color fill resource allows a great deal of tailoring of contour plots. The color fill values, colors and attributes especially for large lists can be cumbersome specified on the command line. Therefore, these resources can be specified in a color fill file. The parameters in this file are the same as those listed in the **color_fill** resource but the multiple parameters are listed on separate lines.

FORMAT

The syntax for each line is:

```
[value:]color[:attrib]
[value:]color[:attrib]
```

...

Where:

- *value* -- A contour value cutoff for the lower end of the contour range.
- *color* -- The color value (either number or name) or a range of color values (either numbers or names). The range is specified as follows:

color-color

This represents a contiguous range of numbers or of color names. For example, **2-4** gives colors **2,3,4**. The values **red-blue** will give all the colors between **red** and **blue** in the color table. A value of **off** means no fills will be used (area is transparent). A color of **end** terminates the list.

- *attrib* -- Any of the color attributes like fill style, line width each separated by a colon listed below:

Attribute	Description
wi=width	Specifies the width of lines. This includes lines used in text and markers. The default value is 1.0 .
st=style	Specifies the style of lines. Possible values are: <ul style="list-style-type: none"> • sol - solid lines (number 1) • dsh - dashed lines (number 2) • lsdsh - long short dashed lines (number 3) • llsdsh - long, long, short dashed lines (number 4) • dot - dotted lines (number 5) • # - a number corresponding to the above styles
fo=font	Specifies the font name. See the font list resource.
fi=fill fp=fill	Specifies the fill pattern. Possible values are: <ul style="list-style-type: none"> • sol - solid fill (number 0) • st - stippled fill (number 1) • ost - open stippled (number 2) • rst - random stipple (number 3) • lst - large stipple (number 4) • vln or - vertical lines (number 5) • drl or // - diagonal lines moving to upper right (number 6) • dlr or \\ - diagonal lines moving to lower right (number 7) • hln or -- - horizontal lines (number 8) • hat or XX - hatched lines (number 9) • 0% - 0 % fill (number 10) • 1% - 11% fill (number 11) • 2% - 22% fill (number 12) • 3% - 33% fill (number 13) • 4% - 44% fill (number 14)

	<ul style="list-style-type: none"> • 5% - 55% fill (number 15) • 6% - 66% fill (number 16) • 7% - 77% fill (number 17) • 8% - 88% fill (number 18) • 9% - 100 % fill (number 19) • ## - a number associated with the above patterns
sc=scale hi=height	Specifies the scale factor. For text, this is the text height. For markers, this is the size of the marker. The default value is 1.0 .
te=expan	Specifies the text expansion factor. This controls how wide text is plotted. An expansion factor greater than 1 results in fat text. A value less than 1 results in thin text. The default value is 1.0 .

EXAMPLES

For example:

```
0:white
10:red
30:blue
100:green
```

Specifies to contour white all areas running from values 0 to 10, red from 10 to 30, green from 30 to 100 and all values above 100 as green. The values less than 0 are not contoured.

```
0:white:st=2
10:red:wi=3
30:blue:fp=14:st=3
100:green:fp=16
```

For color fill contours, the attributes would use fill pattern 14 for the blue areas and fill pattern 16 for the green areas.

Here is a sample **prec.cfl** file:

```
.01:DDMagenta
.05:DMagenta
.1:MMagenta
.175:BMagenta
.25:Blue
.375:LBlue
.5:Green
.75:LGreen
1:Brown
1.5:Yellow
2:Red
3:Lred
4:LGray
```

SEE ALSO

Font File (.fnt)

The font file contains the X and Y coordinates used in drawing text labels and values.

FORMAT

The format of the file is:

```

num
char0 x0 y0 x1 y1 ... 255
char1 x0 y0 x1 y1 ... 255
...
charx x0 y0 x1 y1 ... 254
c0 c1 c2 c3 ...
... c254 c255

```

Where:

- *num* -- The total number of X,Y points in the font. This total includes X, Y (each X Y pair is considered 2) and terminator values.
- *char0* -- The first character in the font. This will be an ASCII character. This is ignored by WXP but is handy for debugging.
- *x0 y0* -- This is a X,Y coordinate pair. The possible values for a draw coordinate are 0 to 127 in X and 0 to 160 in Y. The values for a move coordinate are 128 to 253 in X and 0 to 160 in Y. The character is drawn by connecting X,Y points. There are two operations:

MOVE: This moves, but does not draw to a new X,Y location. A move coordinate is specified by adding 128 to the X coordinate.

DRAW: This draws a line from the previous point to the current one. The X and Y coordinates are taken literally.

The Y coordinate is defined as follows:

```

TOP = 160
CAP = 140
HALF = 85
BASE = 35
BOTTOM = 0

```

Most letters are defined as extending from base to cap. Descenders fall to the bottom value. Some characters such as parentheses () extend above and below the cap and base. The total vertical height is 105 and this value is the divisor for character height.

- **255** -- Specifies that this is the end of this particular character. The terminator on the last character in the font is **254**.
- *c0 c1 c2 ... c255* -- The character lookup table. This gives the offset into the character table for the beginning of each character. This table has 256 values in, one for each possible character in the font. If the character is not defined in the X, Y table, the offset value is **-1**.

EXAMPLES

A sample of the **modern.fnt** file looks like:

```

2221
! 133 140 5 70 133 45 0 40 5 35 10 40 5 45 255
" 128 140 0 105 168 140 40 105 255
# 168 160 5 0 198 160 35 0 133 95 75 95 128 65 70 65 255

```

```

$ 153 160 25 15 173 160 45 15 198 125 60 135 45 140 25 140 10 135
  0 125 0 115 5 105 10 100 20 95 50 85 60 80 65 75 70 65
  70 50 60 40 45 35 25 35 10 40 0 50 255
% 218 140 0 35 153 140 35 130 35 120 30 110 20 105 10 105 0 115
  0 125 5 135 15 140 25 140 35 135 50 130 65 130 80 135 90 140
  198 70 60 65 55 55 55 45 65 35 75 35 85 40 90 50 90 60
  80 70 70 70 255
& 228 95 100 100 95 105 90 105 85 100 80 90 70 65 60 50 50 40
  40 35 20 35 10 40 5 45 0 55 0 65 5 75 10 80 45 100
  50 105 55 115 55 125 50 135 40 140 30 135 25 125 25 115 30 100
  40 85 65 50 75 40 85 35 95 35 100 40 100 45 255
' 128 140 0 105 255
...
~ 128 75 0 85 5 100 15 105 25 105 35 100 55 85 65 80 75 80
  85 85 90 95 128 85 5 95 15 100 25 100 35 95 55 80 65 75
  75 75 85 80 90 95 90 105 255
_ 128 10 50 10 254
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 0 15 24 41 90 149 218 223 244 265 278 287
304 309 320 325 360 369 398 429 440 475 522 531 590 637 658
685 692 701 708 747 802 815 858 895 924 941 954 997 1010 1015
1036 1049 1058 1075 1088 1131 1156 1203 1232 1273 1282 1303 1312 1329 1338
1349 1362 1379 1384 1401 2217 1410 1421 1454 1487 1516 1549 1584 1599 1642
1661 1676 1697 1710 1715 1748 1767 1802 1835 1868 1883 1918 1933 1952 1961
1978 1987 2004 2017 2092 2097 2172 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1

```

The sample file has 2221 values in the character table. The first character is '!'. Here is a summary of its points:

1. move to 5,140
2. draw to 5,70
3. move to 5,45
4. draw to 0,40
5. draw to 5,35
6. draw to 10,40
7. draw to 5,45
8. terminator

The lookup table specifies that characters 0 through 32 are not defined (-1). The space character (32) is automatically defined and need not be in the font file. The first defined character is 33, ! and it starts at value 0. The second defined character 34, " starts at value number 15. The first character has 7 X,Y points and a terminator for a total of 15 values.

SEE ALSO

Font List File (.fnl)

Since font names can be quite long, a font list file is provided to create font aliases. The default font list file is "**wxp.fnl**". This file is used for font aliases even if nothing is specified with the **font_list** resource.

FORMAT

The format of the font list file is:

```
alias font_name
alias font_name
...
```

Where:

- *alias* -- The alias to use when specifying fonts through the color resources "**-cod=green:fo=modern**"
- *font_name* -- The full font name. By default this is a WXP font file but it can be a system font if the font name is preceded by a "#". System fonts are for display only.

EXAMPLES

Here is a sample font list file:

```
mod      modern.fnt
helv     #-adobe-helvetica-bold-r-normal-*-12-*-*-*-*-*-*-*
courier  #-adobe-courier-bold-r-normal-*-12-*-*-*-*-*-*-*
btimes   #-adobe-times-bold-r-normal-*-12-*-*-*-*-*-*-*
times    #-adobe-times-medium-r-normal-*-12-*-*-*-*-*-*-*
school   #-adobe-new century schoolbook-bold-r-normal-*-12-*-*-*-*-*-*-*
wtimes   #Times New Roman:14
warial   #Arial
```

The alias is listed first and the actual font name is listed next. If the font is preceded with a "#", it is a system font and not a WXP font. The first font listed is a WXP font. The next 5 are X11 fonts (Unix) and the last two are Windows fonts. System fonts are generally fixed and won't scale as the window is enlarged. System fonts are not printable. To specify a font, use the color resource: "**-cot=white:fo=warial**" or the plot parameter "**-pp=font:helv**". The default font list file is "**wxp.fnl**" which will be used if it exists. Otherwise, WXP will use the "**modern.fnt**" file or the fonts listed in the **font_list** resource. To specify a different font list file, use "**-fl=fi:new_fonts.fnl**".

SEE ALSO

Map File (.map and .bmap)

The map outline database file contains the latitudes and longitudes of political and geographical boundaries that can be used to draw maps. These files are available in ASCII and binary formats. The ASCII format is available for easy modification. The binary format is for quick drawing. There is a program called **map2bin**, which will convert from one format to the other.

ASCII FORMAT (.map)

The political and geographical boundaries are split up into blocks of continuous line segments. Each block contains a header listing the number of points and the latitude-longitude extent of the block along with the latitude-longitude pairs which form the line segment. The header of each block is formatted as follows:

```
num maxlat minlat maxlon minlon off
lat1 lon1 lat2 lon2 lat3 lon3 ...
num maxlat minlat maxlon minlon off
...
```

where:

- *num* -- The number of latitude-longitude pairs in the block,
- *maxlat* -- The maximum latitude in the block.
- *minlat* -- The minimum latitude in the block.
- *maxlon* -- The maximum longitude in the block.
- *minlon* -- The minimum longitude in the block.
- *off* -- The byte offset from the beginning of the file of the first character of the next block of data.
- *lat1 lon1* -- The first latitude longitude pair
- *lat2 lon2* -- The second latitude longitude pair

The latitude-longitude pairs follow header block. There is a total of *num* pairs which is followed by the next header block.

BINARY FORMAT (.bmap)

The binary map file has the same structure as the ASCII file. These files were created with C **fwrite** (big endian) calls. The header is formatted as follows:

- *XX* -- 2 byte short integer -- The number of latitude-longitude pairs in the block.
- *XXXX* -- 4 byte IEEE floating point -- The maximum latitude in the block.
- *XXXX* -- 4 byte IEEE floating point -- The minimum latitude in the block.
- *XXXX* -- 4 byte IEEE floating point -- The maximum longitude in the block.
- *XXXX* -- 4 byte IEEE floating point -- The minimum longitude in the block.
- *XXXX* -- 4 byte long integer -- The byte offset (from the beginning of the file of the first character of the next block of data (unused in the binary file).

The latitude longitude pairs are written as a continuous stream of binary 4 byte IEEE floating point values.

```
XXXXXXXXXXYYYYYY...
```

- *XXXX* -- 4 byte IEEE floating point -- The latitude of a point
- *YYYY* -- 4 byte IEEE floating point -- The longitude of a point

MAP2BIN PROGRAM

This program converts the ASCII map files to binary or from binary to ASCII. The syntax of the program is:

```
map2bin in_map out_map
```

The program will automatically sense which type of file is being used as input and convert to the other format. If *out_map* is omitted, the output is to standard output.

EXAMPLES

An example of a block from the **wxp.map** file:

```

24 49.00 45.55 -116.92 -124.75 371
48.15 -123.70 48.35 -124.75 47.90 -124.62 47.00 -124.18 46.28 -124.00
46.17 -123.15 46.08 -122.90 45.65 -122.77 45.55 -122.25 45.70 -121.80
45.65 -121.17 46.00 -119.00 46.00 -116.92 46.40 -117.00 49.00 -117.00
49.00 -120.00 49.00 -122.75 48.60 -122.42 48.00 -122.20 47.30 -122.30
47.35 -122.55 47.80 -122.50 48.12 -122.77 48.15 -123.70
14 46.28 42.00 -116.50 -124.55 602
46.00 -116.92 45.60 -116.50 44.48 -117.20 44.30 -117.20 44.15 -116.90
43.80 -117.00 42.00 -117.00 42.00 -120.00 42.00 -122.00 42.00 -124.20
42.83 -124.55 44.00 -124.15 45.00 -124.00 46.28 -124.00
19 49.00 42.00 -111.05 -117.00 903
49.00 -117.00 49.00 -116.05 48.00 -116.05 47.70 -115.75 47.45 -115.75

```

Map Databases

Database	Coverage	Resolution	Outlines
wxp.map	North America	low	Continental, Country, State
wxpstate.map	North America	low	Country, State
cont.map	Global	low	Continental
country.map	Global	low	Country
state.map	United States	high	Continental, State
usa_cnty.map	United States	high	County, State
river.map	Global	high	Rivers
zones.map	United States	high	Forecast zones
DLG	United States	high	Digital Line Graph: Continental, State, County, Rivers, Roads, Misc Political Boundaries

SEE ALSO

- [mapplt](#)

Map List File (.mpl)

The map list file provides a means for specifying complicated map plots. This mimics the **map_file** resource but often it is impossible to specify complex maps on the command line.

FORMAT

The map list file is a list of map files along with flags and attributes:

```
[flag:]map_file[:attributes...]
[flag:]map_file[:attributes...]
...
```

where:

- *flag* -- Toggle flag which determines whether the map is drawn or not. This can be one of the following:
 - *>size* -- Only draw map if domain size, $(ny-1)*dy$, is greater than *size*
 - *<size* -- Only draw map if domain size is less than *size*
 - [*minlat,maxlat,minlon,maxlon,maxdom*] -- Only draw map if domain is within the latitude, longitude range and the domain size is less than *maxdom*. If this flag is specified, no map files are listed on the same line but this flag applies to all subsequent maps up to and the next bracket map range line. If the bracket are empty "[]", this the default case and must be listed last
 - *+* -- Only draw map as an overlay
 - *-* -- Only draw map as an underlay
- *map_file* -- The map file to draw. This is the name of a map file to use or one of the following:
 - **lalo**:*lat[:lon]* -- Draw lat-lon lines at the specified interval. This allows the user to control the order and when lat-lon lines are drawn. If *lon* is omitted, the interval for both is the same.
 - **rf**:*rawfile[-variable]* -- Plots the contents of the raw file This is handy for labeling maps with city and county names.
 - **im**:*image[:back:x:y]* -- This overlays an image on the plot. The image is a GIF image that can be created by WXP or another program and saved in the file *image*. The value of the **image_path** resource is prepended to the filename if a relative path. The *back* specifies a color that will be transparent. The default is **black**. If this plot is a reverse/inverted color (white background), the map image may have to be plotted in black in which case you would use **white** for the transparent color. The *x:y* specifies an offset for the image. By default, it is placed relative to the upper left corner of the plotting window. You can offset this positive which is relative to the upper left corner of negative which is relative to the lower right corner.
 - **if**:*image[:back:x:y]* -- This overlays an image much like **im**: except the image is offset from the upper left corner of the window, not the plot.
 - **mf**:*metafile* -- This plots the content of a metafile.
- *attributes* -- This is a list of attributes to tailor the map:
 - **co**=*color* -- The color the map is drawn in
 - **wi**=*width* -- The line width for the map
 - **fp**=*fill* -- The fill pattern for closed regions.

EXAMPLES

An example of a block from the **cnty.mpl** file:

```
>20:wxp.map
<10:dlg/cnty.bmap:red
>10<20:dlg/nation.bmap:red
>10<20:dlg/state.bmap:red
>10<20:dlg/coast.bmap:red
<10:dlg/nation.bmap:lred:w2
<10:dlg/state.bmap:lred:w2
```

```
<10:dlg/coast.bmap:lred:w2  
<2:dlg/rd_state.bmap:dgray  
<3:dlg/rd_us.bmap:dgray  
<10:dlg/rd_int.bmap:brown  
<3:rf:counties.raw-Name:co=cyan:te=.75
```

SEE ALSO

- [mapplt](#)

Model Lookup File (model.lup)

For gridded data from the forecast models, information is broadcast on a set of grids. Often these grids are saved into output files based on the model and grid type. In some cases, several types of grids are broadcast from a single model. For example, the Eta model has grids that cover the US (211), Alaska (207). In other cases, the grid is broken into subgrids, as is the case with the aviation model. The global grids are broken up into 8 subgrids which must be pieced together to produce the final global grid. In addition, there needs to be a mapping between model type ([model](#) resource) and the file name tags. The "**model.lup**" file serves these purposes.

FORMAT

The syntax of the model lookup file is:

```
model geometry tag[#grid][,tag[#grid]...]
```

Where:

- *model* -- The name of the model used in the **model** resource (**-mo=eta**).
- *geometry* -- The geometry of the grids to be pieced together. If geometry is "1", only one grid is used. If multiple grids are pieced together, this listed the way they are pieced together. For example, "4x2" means that 8 grids will be pieced together with 4 per row and 2 rows. The grids are pieced together from left to right and top to bottom.

```
1234
5678
```

- *tag* -- The file name convention tag to be used in constructing the file name to use. More than one tag can be listed if the grids are to be pieced together (see *geometry*).
- *grid* -- The grid number from the GRIB file (PDB Octet 7). This is used to get specific grids from a file containing more than one grid type.

For more information, see the [forecast model files](#) section of the Users Guide.

EXAMPLES

Here are some examples:

```
eta 1 grib_eta
```

The Eta model grids would be placed in a single file by the ingestor. The model would be specified using "**mo=eta**" and the grids would be searched for in a file with the "**grib_eta**" naming convention (see the name convention file).

```
eta 1 grib_eta#211
eta_ak 1 grib_eta#207
```

This would be used if more than one grid type was saved into a specific file by the ingestor. If the model is "**eta**", then only the 211 grids from the file would be used. If the model is "**eta_ak**", then the 207 grids would be used.

```
avn 4x2 grib_avn_n1w,grib_avn_n0w,grib_avn_n0e,grib_avn_n1e,
grib_avn_s1w,grib_avn_s0w,grib_avn_s0e,grib_avn_s1e
```

This would specify that the **avn** model grids are actually pieced together from 8 separate grids. The geometry is **4x2** and the grids are extracted from files with naming conventions of **grib_avn_n1w**, and **grib_avn_n0w**, etc. Since the piecing process can take time, smaller sub grids can be used:

```
avn_nae 3x1 grib_avn_n1w,grib_avn_n0w,grib_avn_n0e
```

This will only piece together 3 grids from the northern hemisphere rather than piecing together 8. This is useful for AVN plots over North America.

An example of the **model.lup** file:

```

eta      1   grib_eta
eta_ak   1   grib_etak
meta     1   grib_meta
ngm      1   grib_ngm
avn_us   1   grib_avn_us
avn_na   1   grib_avn_na
avn_nh   1   grib_avn_nh
avn      4x2 grib_avn_nlw,grib_avn_n0w,grib_avn_n0e,grib_avn_nle,
grib_avn_slw,grib_avn_s0w,grib_avn_s0e,grib_avn_sle
avn_nhem 4x1 grib_avn_nlw,grib_avn_n0w,grib_avn_n0e,grib_avn_nle
avn_shem 4x1 grib_avn_slw,grib_avn_s0w,grib_avn_s0e,grib_avn_sle
avn_whem 2x2 grib_avn_nlw,grib_avn_n0w,grib_avn_slw,grib_avn_s0w
avn_ehem 2x2 grib_avn_n0e,grib_avn_nle,grib_avn_s0e,grib_avn_sle
avn_nae  3x1 grib_avn_nlw,grib_avn_n0w,grib_avn_n0e
avn_asia 2x1 grib_avn_n0e,grib_avn_nle
avn_eur  3x1 grib_avn_n0w,grib_avn_n0e,grib_avn_nle
avn_sam  2x1 grib_avn_s0w,grib_avn_s0e
mrf      1x2 grib_mrf_nh,grib_mrf_sh
mrf_us   1   grib_mrf_us
mrf_nh   1   grib_mrf_nh
mrf_hi   1   grib_mrf_hi
mrf_nhem 1   grib_mrf_nh
mrf_shem 1   grib_mrf_sh
ruc      1   grib_ruc
ecmwf    4x2 grib_ecmwf#4,grib_ecmwf#3,grib_ecmwf#2,grib_ecmwf#1,
grib_ecmwf#12,grib_ecmwf#11,grib_ecmwf#10,grib_ecmwf#9
ecmwf_nh 4x1 grib_ecmwf#4,grib_ecmwf#3,grib_ecmwf#2,grib_ecmwf#1
ecmwf_tr 4x1 grib_ecmwf#8,grib_ecmwf#7,grib_ecmwf#6,grib_ecmwf#5
ecmwf_sh 4x1 grib_ecmwf#12,grib_ecmwf#11,grib_ecmwf#10,grib_ecmwf#9
sst      2x2 grib_sst#61,grib_sst#62,grib_sst#63,grib_sst#64
sst5     2x2 grib_sst#21,grib_sst#22,grib_sst#23,grib_sst#24
sst1     4x2 grib_sst#37,grib_sst#38,grib_sst#39,grib_sst#40,
grib_sst#41,grib_sst#42,grib_sst#43,grib_sst#44

```

SEE ALSO

- [griblook](#)
- [grbcalc](#)
- [ingest](#)

Model Name File (mod_name.lup)

The model name database file contains a list of model numbers from the GRIB files and a name to use in labels for WXP products. The model names change on a regular basis so rather than hard coding the model names into the program, the model lookup file provides the means for updating the model name list and to tailor how the model name is displayed on WXP products.

FORMAT

The model name file is formatted as follows:

```
[src:]model name
[src:]model name
...
```

where:

- *src* -- The optional source/center ID from the GRIB product (Octet 5 of PDB, ie ECMWF=98, NCEP=07)
- *model* -- The model ID from the GRIB product (Octet 6 of PDB)
- *name* -- The name to use for the model (ie Eta, NGM, AVN)

EXAMPLES

An example of the **mod_name.lup** file:

```
0 OBS
# WXP derived fields
1 WXP
# Significant Wind Wave model
10 SWW
# Snow Cover Analysis
25 SCA
# European Center for Medium range Weather Forecasting model
98:30 ECMWF
98:31 ECMWF
98:40 ECMWF
98:111 ECMWF
98:122 ECMWF
98:131 ECMWF
98:141 ECMWF
98:151 ECMWF
98:152 ECMWF
98:161 ECMWF
98:183 ECMWF
98:184 ECMWF
# Nested Grid Model
39 NGM
239 NGM MOS
# Global Optimal Interpolation analysis
43 GOI
# Sea Surface Temperature analysis
44 SST
# Spectral 24 wave model
45 S24
# Spectral 30 wave model
46 S30
# Limited Fine Mesh model
53 LFO
```

```
# Regional Optimal Interpolation analysis
64 ROI
# Medium Range Forecast model
69 AVN
77 AVN
78 MRF
80 MRF
81 AVN
82 MRF
278 MRF MOS
# ETA Forecast model
83 Eta8
84 Eta4
85 Eta3
89 Eta
289 Eta MOS
# RUC Forecast model
86 RUC
# US Navy
58:71 Navy
```

SEE ALSO

- [griblook](#)
- [grbcalc](#)
- [ingest](#)

Name Convention File

The file name convention file links the **file tags** to the actual file names. The name convention specifies how date and data type are to be formatted into the filename, making it easier for programs to find data files that are current or at a specific time.

FORMAT

Each line in the data file contains a specific file name convention for a particular type of data. The line contains a **name convention tag** followed by the **name convention string**.

```
tag filename [headerfilename]
```

Where:

- *tag* -- The file name tag that is used in the **in_file** and **out_file** resource.
- *filename* -- The name convention for the data file name.
- *headerfilename* -- The name convention for the header file name (optional).

File Name Tags

The tag is important as most WXP programs use specific tags to find specific name conventions. The tag itself has two parts: a data type followed by a file type. For example, surface data saved by the ingestor would have a tag of **sfc_dat** which is input to the surface conversion program. The input and output tags for surface data are:

sfc_dat -> **sacvt** -> **sfc_cvt**, **sfc_cvt_wxp** (WXP format), **sfc_cvt_cdf** (netCDF format).

sfc_cvt -> **sfcwx** -> **sfc_raw** (raw output), **sfc_grd** (grid output), **sfc_grd_wxp** (WXP grid output), **sfc_grd_cdf** (netCDF grid output)

File Name Conventions

The **name convention string** is generally a combination of the file path and a name convention based on date and file type. The string uses a wildcard substitution system much like the formatting of the **printf** function. Anything not part of a wildcard is taken literally. Here is a list of the wildcards:

Wildcards	Description
PATHS	
%F	file_path resource
%D	data_path resource
%C	con_path resource
%G	grid_path resource
%R	raw_path resource
%I	image_path resource
%W	watch_path resource
DATE	
%Y	Current year (1900-)
%y	Current year (00-99)
%B	Current month (JAN-DEC)
%b	Current month (jan-dec)
%m	Current month (01-12)
%j	Current Julian day (0-365)
%d	Current day (01-31)
%h	Current hour (00-23)
%n	Current minute (00-59)
PRESET VALUES	

%r	region
%l	vertical level
%f	forecast time
%v	variable
%x	model
%p	program name
%i	loop/frame index
%e	tag extension

To handle multiple hour/minute files, a number can be added to the wildcard. A **%6h** represents a data file created once every 6 hours or a 6 hour compilation file. A **%5n** specifies that a data file is created once every 5 minutes.

EXAMPLES

Here are some examples:

```
sfc_dat      %D/%y%m%d%h_sao.wmo
sfc_cvt_cdf  %C/%y%m%d%h_sao.nc
sfc_cvt      %C/%y%m%d%h_sao.wxp
sfc_raw      %R/%y%m%d%h-%v_sao.raw
sfc_grd_cdf  %G/%y%m%d%h-%v_sao.nc
sfc_grd      %G/%y%m%d%h-%v_sao.grd
```

Not all of these need to be set for each type. In general only the "**sfc_dat**" and "**sfc_cvt**" need to be set.

Name Convention String	Sample Filename
%D/%y%m%d%h.sao	/home/wxp/data/97092714.sao
%C/%12h%m%d%y.uac	/home/wxp/convert/12092797.uac
%G/%y%m%d%12h_%f_%l_%v.grd	/home/wxp/grid/97092712_h24_500_temp.grd

To get more information on the use of name conventions, check out the [name convention](#) section of the Users Guide.

SEE ALSO

Parse Lookup File (parse.lup)

For most WXP program, they can find data files by use of the name convention tag. For parsing, there is an additional method that simplifies the process. When the parse program parses for a WMO header, it searches the parse lookup file "**parse.lup**" to cross reference headers to name convention tags. This eliminates the need to specify the **in_file** file name tag.

FORMAT

The syntax of the file is:

```
pattern      tag
```

Where:

- *pattern* -- The pattern to match against the requested WMO header. This can be a regular expression.
- *tag* -- The file name convention tag to use in determining the file to parse. The lookup search returns the tag of the first match in the file. That name convention is used to setup the filenames to search for the desired product.

EXAMPLES

Here is a sample parse lookup file:

```
FO      mos_dat
FT      term_dat
W       sev_dat
F       for_dat
C       cli_dat
S       sfc_dat
UA      pirep_dat
UB      pirep_dat
UD      pirep_dat
U       upa_dat
AC      sev_dat
A       sum_dat
FO      mos_dat
```

In this case, all products whose WMO header starts with "W" will be searched for in the files with the tag "**sev_dat**".

SEE ALSO

- [parse](#)

Region File (.reg)

WXP uses a default region file called "**wxp.reg**". In this file, there is a list of all regions that can be used in WXP and their appropriate **plot_domain** specifications. Often, the default set of regions does not satisfy the users needs so new regions can be added to this file if needed.

FORMAT

The syntax of this file is similar to the variable and menu files:

```
abbr      name      toggle      plot_domain
```

Where:

- *abbr* -- The region abbreviation to be used in specifying the region with the **plot_domain** resource.
- *name* -- The full name of the region to be used in the region menu. No spaces are allowed but underscores "_" can be used in their place.
- *toggle* -- This is a menu toggle which defines when the menu item will appear. This can be one of the following:
 - **1** -- the item always appears
 - **0** -- the item never appears.
- *plot_domain* -- The plot domain specification. This cannot be a region abbreviation but must be the full domain specification such as "**39,-97,2.3**".

This file also controls which regions will show up in the region menu for appropriate WXP programs. Also, dashes can be placed in each value to delimit a separator. This is useful for organizing items in the region menu.

EXAMPLES

An example of a block from the **wxp.reg** file:

```
us      Contiguous_US      1  39,-97,2.3
ne      New_England      1  42,-76,.9
at      Atlantic      1  37,-82,.9
se      Southeastern      1  31,-88,1.1
mw      Midwestern      1  43,-93,1.1
sp      Southern_Plains      1  32.5,-100,1.2
nw      Northwestern      1  44,-112,1
sw      Southwestern      1  37,-112,1
--      -----      1  -----
wcan    Western_Canada      1  55,-110,1.3
ecan    Eastern_Canada      1  53,-75,1.3
ncan    Northern_Canada      1  70,-100,1.5
namer   North_America      1  50,-100,49,33,2.2
ngm     NGM      0  40,-101,2.9
nhem   North_Hemisphere      0  90,-90,25,25,8
shem   South_Hemisphere      0  -90,-90,25,25,8
hi     Hawaii      0  21,-157,.6
ak     Alaska      0  65,-140,1.5
col    Colorado      1  39,-106,.8
ind    Indiana      1  40,-86,.3
```

SEE ALSO

Resource File (.wxpdef or wxp.def)

The resource file contains a list of resource defaults used by all WXP programs.

FORMAT

Each line in the resource file contains a resource and its default value:

```
resource_name: default_value
```

Resources can be tailored by program, by name or set globally:

- **Global** - these are used by all programs and are listed at the top of the resource file. A global resource is used unless a named resource is specified to override the global resource. These are specified by preceding the resource name with an asterisk:

```
*resource_name: default_value
```

For example, to set the global `data_path` resource:

```
*data_path: /home/wxp/ldm/wxp/ddplus
```

- **Named Program** - these are used only by a specific program. These are specified by preceding the resource name with the name of the program and a period:

```
WXP_program.resource_name: default_value
```

A program specific resource can be set:

```
grbcalc.data_path: /home/wxp/model
```

This value will override the global **data_path** resource.

- **Named** - these are used when special cases need to be defined (e.g. working with synoptic data rather than SAO/METARs). These are specified by preceding the resource name with a specific name chosen by the user:

```
special_name.resource_name: default_value
```

For example:

```
ngm.model: ngm
```

To enable the name, the user must specify the name on the command line when the program is executed:

```
grbcalc -name=ngm ...
```

Complex Resources

Named resources can have complex names in order to organize defaults. A wildcard character "*" can be used to use complex resource defaults in more than one case. For example the following resource structure is desired:

```
grbcalc
  data_path: /home/wxp/model
  plot_domain: ngm
```

```

ngm
  model: ngm
  prec
    color_table: rainbow.clr
    color_fill: prec.cfl
    color_cmap: white
    variable: prec
    plot_type: cf
    level: snd

```

A full resource name would be **grbcalc.ngm.prec.level** but we want to carry over all the resources from grbcalc and ngm. These can be specified with a combination of absolute and wildcard resource specifications.

```

grbcalc*data_path: /home/wxp/model
grbcalc*plot_domain: ngm
*ngm*model: ngm
*ngm.prec.color_table: rainbow.clr
*ngm.prec.color_fill: prec.cfl
*ngm.prec.color_cmap: white
*ngm.prec.variable: prec
*ngm.prec.plot_type: cf
*ngm.prec.level: snd
The named resource will be specified as:
grbcalc -name=grbcalc.ngm.prec

```

and this will use all the resources listed above. This way resource files can be simplified and changing one resource can be propagated quickly to the other complex resources.

There is a shortcut so the program name does not need to be specified every time:

```
grbcalc -name+=ngm.prec
```

and the program name is automatically prepended.

Conditional Resources

In some cases, resources may need to be changed in rare occasions. For example, a data feed is down and an alternate feed needs to be used. An example of this is:

```

IF NOAAPORT
grbcalc*data_path: /data/model
ELIF WXP
grbcalc*data_path: /home/wxp/model
ELSE
grbcalc*data_path: /home/wxp/data
END LDM

```

In this case, the first line is used if NOAAPORT is defined, otherwise the second if WXP is defined or the third otherwise. You can define NOAAPORT in two places. You can add a line to the resource file:

```
DEF NOAAPORT
```

This carries over to all subsequent resource files. You can undefine a tag with:

```
UNDEF NOAAPORT
```

The more common method is to add the definition to the **wxpdefault** environment variable:

```
setenv wxpdefault +noaaport:~devo/.wxpdef
or
grbcalc -default +noaaport:~devo/.wxpdef
```

EXAMPLES

Here is a sample resource file:

```
#
# Global resources
#
*file_path: /home/wxp/etc
DEF GATEWAY
IF GATEWAY
*data_path: /data1/data
*watch_path: /data1/data
*name_conv: name_conv
ELSE
*data_path: /home/wxp/data
*watch_path: /home/wxp/data
*name_conv: name_conv
END
*con_path: /home/wxp/convert
*con_path: /data1/convert
*grid_path: ~/grid
*raw_path: ~/raw
*image_path: ~/image
*background: def
*color_table: rainbow.clr
*color_front: lblue:w4,lred:w4,lblue:w4,lred:w4,lmagenta:w4,brown:w4
#
# Satellite resources
#
IF GATEWAY
xsat*data_path: /data2/sat
ELSE
xsat*data_path: /home/wxp/sat
END
xsat*input: awip
xsat*map_file: cont.map,wxpstate.map
xsat*color_fill: 0-50
xsat*color_table: sat.clr
# GOES 8 Images
xsat.goes8.input: unisat
xsat.goes8.in_file: wxmax
xsat.goes8.data_path: /wxsvr3/goes_8/dir_105
#xsat.goes8.grid_domain: go8fd
# GOES 9 Images
xsat.goes9.input: unisat
xsat.goes9.in_file: wxmax
xsat.goes9.data_path: /wxsvr3/goes_9/dir_260
xsat.goes9.grid_domain: go8fd
```

For more information on resource files, check the [resource file section](#) of the Users Guide.

SEE ALSO

Satellite Enhancement File (.enh)

The satellite enhancement database file is very similar to the color fill database file. The enhancement file is specified with the **color_fill** resource.

FORMAT

The syntax for each line is:

```
[value:]color[%Dcolor]
[value:]color[%Dcolor]
...
```

Where:

- *value* -- A image value cutoff for the lower end of the color range. The color values are image pixel values ranging from 0 to 255. The value may be specified as a image pixel value or as a infrared temperature based on the GOES calibration scheme. The temperature is specified by preceding the value by a "c". For example, "c-25" specifies the image pixel value corresponding to -25 Celsius. This way infrared images can be enhanced based on the temperatures the image represents.
- *color* -- The color name or a range of color values (either numbers or names). The range is specified as follows:

color-color

This represents a contiguous range of numbers or of color names. For example, **2-4** gives colors **2,3,4**. The values **red-blue** will give all the colors between **red** and **blue** in the color table. A color of **off** can be used to blank out (make transparent) parts of the image.

- *Dcolor* -- The colors can be specified with dither value *D* and *color*. Dither colors are specified by using the following syntax:

%6white

The original or background color is specified followed by a "%", then the dither percent value and the dither or foreground color. The above example uses the 3x3 dither matrix and gives possible dither values of 0 through 9. The result is a 6/9 dither of white over black. Using 0 for the dither value gives the original color. Using 9 for the value gives the only dither color. This is somewhat restrictive for monochrome screens, so a second dither scheme is provided which is a 5x5 dither matrix. This gives a total of 26 possible dither values which are specified by lower case alphabetic characters "a" through "z". The value "a" gives the original color and "z" gives the dither color. For example:

%mwhite

gives a 12/25 dither.

The color fill file or more appropriately the enhancement file has an additional line at the beginning of the file that names the enhancement. Here is the MB enhancement file (**sat_mb.enh**):

```
MB
0:black
c29:1
2-9
c6:10
11-15
c-31:16
```

c-32:cyan
c-43:lcyan
c-54:lblue
c-60:blue
c-64:green
c-70:lgreen
c-76:brown
c-81:yellow
c-90:black

SEE ALSO

- [xsat](#)

Shell Menu File (wxp.menu)

The WXP User's shell uses a menu to select programs and options. In order to make the shell more extensible, the menu items and the menus themselves can be changed at any time. This is done by modifying the "**wxp.menu**" file located in the WXP database directory (see [file path](#)). This file lists all the menus, the labels used to describe each action and the command to run when the item is selected.

FORMAT

The format of the file is as follows:

type label command

Where:

- *type* -- The type of menu item which is either a "**M**" for menu or "**I**" for item. The menu type denotes the beginning of a menu listing. The subsequent lines will be item lines until the next menu line with those items being displayed as part of that menu.
- *label* -- The label is a single string that is used to describe the menu item in more detail than just the command. This string cannot contain white space so any white space characters must be replaced by underscores "_". The number of characters is limited to 40.
- *command* -- The command is simply the command to run for that item. It is the command and any command line parameters. If the item is of type menu, the command becomes the short name of the menu and is used with the "**menu name**" command to go to that menu. If the menu item is a reference to a submenu, the command will be "**menu name**" where name is the short name listed with each menu. The number of characters is limited to 50.

EXAMPLES

Here is a sample **wxp.menu** file:

```
M   Main                main
I   Parsing_Programs   menu parse
I   Plotting_Programs  menu plot
I   Contouring_Programs menu contour
I   Meteorological_Calculations menu calc
M   Parse_Data         parse
I   Parse_Raw_Surface_Data sa_parse
I   Parse_Surface_Data  sa_parse -if=cvt
I   Parse_Series_of_Surface_Data sa_parse -nh=-6 -if=cvt
I   Parse_Upper_Air_Data ua_parse
I   Parse_MOS_Air_Data  fo_parse
I   Parse_Forecasts    forecast
I   Parse_Text_Data    parse
M   Plot_Data          plot
I   Plot_Surface_Data  sfcwx
I   Plot_Surface_Meteograms statlog
...

```

The first menu is always the main menu. Lets break this down:

- **M** - denotes menu type
- **Main** - the label to be used with the menu will be "Main"
- **main** - the short name of the menu

In this menu, all the items in this menu go to submenus. Let's look at the first item:

- **I** - denote item type
- **Parsing_Programs** - the label used in the menu will be "Parsing Programs"
- **menu parse** - this runs the menu command to go to the "parse" submenu

Now if we look at the parse submenu, each item now runs a program. In some cases, you will notice that command line parameters are specified. This can be handy if you need to specialize an program for a particular application. This is especially handy in a classroom situation where special commands may need to be easily accessed through the WXP menu.

SEE ALSO

- [wxp](#)

Symbol File (.smb)

Symbols files are used to plot various symbols that cannot be plotted with the text fonts. This is especially handy for weather, cloud and pressure tendency symbols.

FORMAT

The symbol file contains a set of symbols defined by a simple character string such as "RW-". Following the string is a set of X,Y points which describe the symbol. The points can define a line or an area to fill. The format is:

```
str0 x0,y0 x1,y1 x2,y2 ... D
      x0,y0 x1,y1 x2,y2 ... D
      ... $
str1 x0,y0 x1,y1 x2,y2 ... D
...

```

Where:

- *str0* -- The string that will be used to designate the symbol (ie "TRW-" for thunderstorms)
- *x0,y0 x1,y1 ...* -- The coordinates used to draw the symbol. These form a continuous line/polygon. The values for X and Y are floating point numbers between 0 and 1.
- *D* -- The draw command
 - **L** -- Draw a line between the listed points
 - **F** -- Fill the area defined by the list of points
- **\$** -- Terminates the draw commands for that symbol.

EXAMPLES

Here is a sample of the **wx.smb** file:

```
T .7,.8 .4,.5 .7,.2 .5,.2 L
  .7,.2 .7,.4 L $
RW- .4,.7 .5,.7 .5,.6 .4,.6 .4,.7 L
    .3,.5 .6,.5 .45,.2 .3,.5 L $
RW+ .4,.7 .5,.7 .5,.6 .4,.6 .4,.7 L
    .3,.5 .6,.5 .45,.2 .3,.5 L
    .35,.4 .55,.4 L $
RW .4,.7 .5,.7 .5,.6 .4,.6 .4,.7 L
    .3,.5 .6,.5 .45,.2 .3,.5 L
    .35,.4 .55,.4 L $
R+ .2,.6 .3,.6 .3,.5 .2,.5 .2,.6 L
    .6,.6 .7,.6 .7,.5 .6,.5 .6,.6 L
    .4,.8 .5,.8 .5,.7 .4,.7 .4,.8 L
    .4,.4 .5,.4 .5,.3 .4,.3 .4,.4 L $

```

WXP comes with several predefined symbol files:

Symbol File	Description
wx.smb	Standard weather symbols (SAO format)
pwx.smb	Synoptic present weather symbols
ptend.smb	Pressure tendency symbols
cl.smb	Low cloud symbols
cm.smb	Middle cloud symbols
ch.smb	High cloud symbols
sev.smb	Severe weather symbols

SEE ALSO

Unit Conversion File (unit.lup)

WXP offers the ability to change the units of a variable prior to plotting. Each variable has a specific unit associated with it when it is read in from the raw or converted data file. When a new unit is specified, the **units.lup** file is searched for a combination of the old and new units and performs the appropriate conversion.

FORMAT

Each line in the file represents a different unit conversion. The format of each line is:

```
old_unit  new_unit  conversion
```

Where:

- *old_unit* -- The original units (ie **F**)
- *new_unit* -- The final units (ie **C**)
- *conversion* -- The conversion factor to use (ie ***.55555-17.7777**)

The old and new units are specified as character strings typical for those units such as "m/s" and "C". The units are case insensitive. Once the conversion is done, the variable now has the new units label attached to it which is carried to new computations.

The conversion is a "y=mx+b" setup. The slope is preceded by a "*" and the intercept is preceded with a "+" or "-". Only one of the two are necessary. If the slope is missing, it is assumed to be 1 and the intercept is assumed to be 0. For example a conversion from Fahrenheit to Celsius (**C=.55555*F-17.7777**) would look like:

```
F      C      *.55555-17.77777
```

The units file is then a set of possible conversions. More may be added if needed. If a conversion is not listed, the units are changed but the value remains the same.

EXAMPLES

A sample of the file is as follows (comment lines are preceded with a "#"):

```
#
#  Temperature
#    F - fahrenheit
#    C - Celsius
#    K - Kelvin
#
F      C      *.55555-17.77777
F      K      *.55555+255.37222
C      F      *1.8+32
C      K      +273.15
K      F      *1.8-459.67
K      C      -273.15
dF     dC     *.55555
dF     dK     *.55555
dC     dK     *1
dC     dF     *1.8
dK     dC     *1
dK     dF     *1.8
```

NOTE: there is a distinction between C and dC. dC values represent difference values such as dewpoint depression or lifted index. The conversion to difference in K (dK) would be different from the conversion of C to K. This is reflected in this notation.

SEE ALSO

Variable File (.var)

Most plotting is done by specifying a variable to plot. The variable is then used by the program to select data from the input data file. In general, the type of output is specified by resources such as [plot type](#), [color data](#), or [plot format](#). In its simplest form, the variable file acts just the same as the time and level menu files. The biggest difference is the last column which represents the variable information. This column allows for variable aliasing, composite plots and overlay products.

Each program that uses the **variable** resource and/or has a variable menu has its own **.var** file. For example, **sfcwx** has a variable file named **sfcwx.var**. The program then parsing this file to determine how to process each variable request.

FORMAT

This file is a list of variables and each variable has a list of commands which specify which variable to plot and how to plot it. The syntax of each line in the file is:

```
Abbrev      Name          Toggle      Definition
```

Where:

- *Abbrev* -- This define an abbreviation which is used to define the variable for use in the [variable](#) resource.
- *Name* -- this is the text to be used in the menu listing. No spaces are allowed but underscores "_" can be used in their place.
- *Toggle* -- this is a menu toggle which defines when the menu item will appear. This can be one of the following:
 - **1** -- the item always appears
 - **0** -- the item never appears. This is not useful in a menu file but is in a variable file.
 - **mo=model** -- a model type. This is the string specified by the [model](#) resource.
 - **ft=time** -- a forecast time. This is a string either specified by the [time](#) resource or at the time menu prompt
 - **le=level** -- a vertical level. This is a string either specified by the [level](#) resource or at the level menu prompt

Wildcard characters are permitted. Each of the above strings can be logically and'd or or'd.

. or ?	match a single character
*	match any character
[<i>letters</i>]	match a single character from the set.
[<i>^letters</i>]	match any character except those from the set.
<i>str</i>	match string
~ <i>str</i>	true if doesn't match string
<i>str1 str2...</i>	match a set or strings (logical or). Pipe " " separates strings
<i>str1&str2...</i>	match all strings (logical and). Ampersand "&" separates strings
_	underscore matches a space.

- *Definition* -- This defines which variable will be plotted and how it is to be plotted. The syntax of the definition is:

```
time:level:variable:units:attributes...
```

The syntax of this is explained below.

A simple entry will look like:

```
temp      Temperature          1 anal:sfc:temp  [F]
```

where:

- **temp** -- is the name used for the variable resource "**-va=temp**".
- **Temperature** -- is the menu label which will show up in the variable menu
- **1** -- specifies that temperature will always appear in the menu
- **anal:sfc:temp[F]** -- is the variable definition. This states that an analysis of surface temperature in Fahrenheit will be plotted.

Here is a sample variable file.

```
temp      Temperature      le=~snd&le=~sfc  temp      [C]
stemp     Temperature      le=sfc           :2m_ag:temp  [C]
t_rh      Temp_Rel_Hum     0               temp,rhum
dewp      Dewpoint        le=~snd&le=~sfc  dewp(t_rh)  [C]
sdewp     Dewpoint        le=sfc           :2m_ag:dewp(t_rh) [C]
wind      Winds           le=~snd&le=~sfc  wind<uwnd,vwnd> [m/s:wind]
swind     Winds           le=sfc           wind<:10m_ag:uwnd,:10m_ag:vwnd> [m/s:wind]
wind8     Winds           0               wind<:850:uwnd,:850:vwnd> [m/s:wind]
wind3     Winds           0               wind<:300:uwnd,:300:vwnd> [m/s:wind]
wdir      Wind_Direction  le=~snd         dir(wind)   [deg]
wspd      Wind_speed      le=~snd         spd(wind)   [knt]
```

Variable Definitions

The variable definition is a quite powerful tool for defining variables, aliasing variables, generating composite, overlay and multipanel plots. The syntax of the definition is:

```
time:level:variable[units:attributes...]
variable[units:attributes...]
```

The sections of the definition are:

- **time** -- This is the forecast time that the variable is valid. This uses a standard time reference (see [time](#) resource)
- **level** -- This is the level that the variable is valid. This uses a standard level reference (see [level](#) resource)
- **variable** -- This is the variable to plot. This can either be a variable defined in the program (internal variables), in the file (GRIB variables) or a reference to another variable in the variable file (variable aliasing).
- **units** -- This is the units to plot the variable in. The program will recompute the variable with the new units prior to plotting.
- **attributes** -- This defines how the variable will appear on the plot. This defines the plot type, colors, etc.

The variable definition can be up to 1000 characters long, can include spaces for readability and can span lines (as long as "\" is the last character of the line). Also, the time and level can be omitted, assuming program defaults (normally the values of the **time** and **level** resources).

The definition can also specify functions, vectors and composite plotting.

The resulting plot will be labeled based not on the menu listing but on the variable to plot. This is looked up in the **variable.lup** file which should cross-reference the variable name "**temp**" with a full name such as in "**Temperature**". The program will then tack on the units to produce a final label of "**Temperature (F)**".

Simple Variables

The default output of a variable is integral. In some cases like altimeter setting, plotting to 2 decimal places is required. This can be established using a plot attribute:

```
alt      Altimeter_Setting      1 alt  [:%.2f]
```

The output format uses a standard C printf floating point format.

In some cases, the data might be not numeric. This is the case with cloud cover which can be a string such as "**B**" for broken. To plot the data rather than the value, specify a plot type of data:

```
cldcv Cloud_Cover          1 cldcv [:data]
```

It is also possible to plot the cloud cover as a symbol. WXP provides a couple of default symbols such as cloud cover and wind barbs. To plot symbols, just specify a different plot type:

```
cldcv Cloud_Cover          1 cldcv [:cloud]
```

In some cases, WXP does not provide symbols directly. To handle this, there is a symbol file which cross-references the data with a list of drawing commands. In one case, present weather, there is a plot type (**wx**) that will read in and use the symbol file **wx.smb**. In other cases, it must be explicitly specified. Notice, the plot type is still "**wx**" but that a different symbol file (or weather file) is specified.

```
ptype Pressure_Tendency    0 ptend [mb/3hr:wx:wf=ptend.smb]
```

Composite overlays

In some cases, it is nice to be able to define new plot types as composites of several variables. The variable definition then becomes a list of variables, separated by commas.

```
lmhcd Clouds                0 city      [:mk=pnt],\
                             lclcd     [:wx:wf=cl.smb:xy=0-.5],\
                             mclcd     [:wx:wf=cm.smb:xy=0+.5],\
                             hclcd     [:wx:wf=ch.smb:xy=0+1]
```

This specifies a set of variables to plot. First, the city location will plot as a point (**mk=pnt** attribute). Then, the low, medium and high clouds will plot as symbols, each using a specific symbol file which cross-references the value to a set of plotting commands. In order to keep each plot from overlapping, an x,y offset is applied to the plot. The offset is normalized so that 1 is roughly the vertical size of text plotted on the screen.

The problem with the above plot is that the label that appears on the plot lists all of the variables. It would be nice to have a composite plot with only one label like "All Data". This is accomplished by enclosing the variables in braces "*var{var1,var2,...}*". Only the first variable *var* is looked up in the **variable.lup** file. In this case, it will plot "**All data**".

```
all      All_Data            1 all{\
                             +temp      [F :ul],\
                             +dewp      [F :ll],\
                             :sl:+pres  [mb:data:ur],\
                             +wbrbc     [ :cbarb],\
                             +cldcv     [ :cloud],\
                             +wx        [ :wx:cl]}
```

This will plot temperature upper left (attribute **ul**), dewpoint lower left, sea level pressure upper right, wind barbs (plot type **cbarb** plots a barb but allows for a cloud cover symbol), cloud cover symbols (plot type **cloud**) and present weather (plot type **wx**). The plusses in front of the variable name will be explained later.

When using this with the **statlog** program, composite plotting can also be used to put more than one variable on a graph.

```
all      All_Data            1 temp {temp,dewp} [F :plot:sz=.18],\
                             extt      [F :value],\
```

```

wx          [ :wx],\
snwdp      [in :value],\
prec       [in :value:%.2f],\
vis        [mi :value],\
wgst       [knt:value],\
wind { \
  wind     [knt:cbarb],\
  cldcv    [% :cloud] },\
cld        [ft :plot:sz=.14],\
cldcl     [100_ft:data],\
pralt     [mb :plot:sz=.14]

```

In this case, the temperature and dewpoint are plotted on the same graph and the attribute information "[F:plot:sz=.18]" refers to the plot and not each individual variable. A new plot type is introduced. "plot" specifies that this will be a plotted graph. There is one other example of compositing. The wind section includes both **wind** and **cldcv** which will produce a standard cloud cover and wind barb. If this is not specified, the cloud cover and barbs will plot on separate lines.

Overlay plots

Overlay is the ability to put two or more plots on the plot and the same time. This is often used with contour plots but can be plotted data over a contour plot:

```

comp1 Composite_1      0 temp    [:cf:in=5],\
                        strm     [:co=black],\
                        cldcv    [:co=black]

```

Again, multiple variables are specified separated by commas. In this case, temperature is contoured (plot type color fill **cf**) with an interval of 5, overlaid with streamlines, colored in black and cloud cover symbols in black.

Complete plots can be specified but the more complex the plot, the more control is needed. In some cases, the map needs to be specifically plotted:

```

cpres Composite_Surface le=snd prec [in:cf:cof=prec.cfl:bar:labs],\
map      [ :co=red],\
slp     [mb:ln:co=lcyan:in=4],\
thk     [m :ln:cof=thick.cfl:co=off:cb=5400:in=60]

```

In this case, 4 variables are being plotted.

1. **prec** -- quantitative precipitation in inches, plotted as a color fill contour (**cf**), using a specific color fill file which lists the values and colors (**cof=prec.cfl**), labeling the plot with a color bar (**bar**) and specifying to use short text labels in upper left (**labs**).
2. **map** -- overlay the map in red (**co=red**)
3. **slp** -- overlay the sea level pressure plotted in millibars, using light cyan (**co=lcyan**) line contours (**ln**) at an interval of 4 (**in=4**).
4. **thk** -- overlay 1000-500 mb thickness in meters, plotted as line contours, line styles defined in the **thick.cfl** file (**cof=thick.cfl**), colors turned off to force use of thick file (**co=off**), contour base set to 5400 to make sure solid line appears at value 5400 (**cb=5400**) and a contour interval of 60 (**in=60**).

This example is also using variable aliasing. The variables **prec**, **slp**, and **thk** are defined elsewhere in the variable file. This will be discussed later.

Plotting Attributes

For a full list of possible attributes, see the following pages of the Users Manual:

- [Data Plotting](#)

- [Gridding and Contouring](#)
- [Vector Plotting](#)

Variable Aliasing

In order to simplify the variable file, variable aliasing is possible. What aliasing does is allow the user to specify a variable that is defined elsewhere in the variable file. Taking the above example, the variable **prec**, **slp** and **thk** are actually:

```
prec  Quant_precipitation  le=snd  prec          [in]
slp   Sea_level_pressure   le=snd  :sl:pres      [mb]
thk   1000-500_mb_thickness le=snd  :p1000-500:thick (:500:hght:m,:1000:hght:m)[m]
```

The variable in this case is also **prec** but if the variable abbreviation matches the variable type, no further searching is done. In the case of sea level pressure, it is actually pressure at sea level. For thickness, this is actually 1000 mb height and 500 mb height run through the function **thick**. That function will just difference the two grids. The function **thick** has a specific level designation in order to format the label properly. The function name is looked up in the **variable.lup** file to get a full label. The resulting label would be:

```
1000-500 mb Thickness (m)
```

In some cases, aliasing can have side effects. To prevent aliasing, add a plus "+" in front of the variable name. This forces the program to immediately search internal variables rather than a recursive search of the variable file. This also clarifies the difference between aliases and real variables.

```
wdir  Wind_Direction      1 +wdir      [deg]
```

In this case, "**wdir**" can be used as an alias elsewhere in the file but "**+wdir**" tells the program to use the internal variable "**wdir**" and not the alias.

Vectors

In some cases, grids need to be grouped in order to represent vector quantities. This is done by enclosing the components in "*var*<*var1,var2*>". The variable *var* is only used to name the plot. The variables *var1* and *var2* define the X and Y components of the vector. This is handy for defining wind as a vector quantity.

```
wind  Wind_vectors        le=~snd&le=~sfc wind<uwndg,vwndg> [m/s:wind]
```

This defines a wind vector based on grid relative U and V wind components. The alias **wind** can then be used elsewhere to represent these components. For example, convergence:

```
conv  Convergence        le=~snd      conv(wind)          [10^-5_/s]
```

This will apply the function **conv** (convergence) to the grid relative U and V wind components. This simplifies the use of vector quantities and makes the variable file more readable.

Functions

Often simple variables do not cover all the possible variable types. In the case of grids, functions can be applied to create new variables.

```
tadv  Temperature_Advection le=~snd      adv(wind,temp:K)    [10^-4_K/s]
```

The advection function **adv** takes 3 parameters: U, V and a parameter to advect. The U and V are represented by the variable **wind** which is a vector field defined elsewhere in the variable file. With many of the functions, the program will make an attempt to produce a label. In the case of advection, it produces a label like "**Surface Temperature Advection (10^-4_K/s)**". In other cases, the name must be forced. For example, the advection function could have been listed as "**adv-tadv**". This will run the **adv** function but it will use the variable **tadv** to lookup a label from the **variable.lup** file.

Some functions can get quite elaborate:

```
p10ht 1000_mb_height 0 :p1000:mul-hght (sub (slp:mb, #1000), #8.4) [m]
```

This computes an estimated 1000 mb height based on sea level pressure. This can now be used in a pseudo-thickness computation:

```
pthk 1000-500_mb_thickness 0 :p1000-500:thick (:500:hght:m, p10ht) [m]
```

Here is an example of wind shear:

```
wind8 Winds_850 0 wind<:850:uwnd,:850:vwnd> [m/s:wind]
wind3 Winds_300 0 wind<:300:uwnd,:300:vwnd> [m/s:wind]
shr83 Shear 0 vdiff(wind3,wind8)
mshr83 Mag_Shear 0 mag(shr83)
```

In this case, a shear value is computed. The variable **shr83** is a vector quantity that can be plotted as vectors or streamlines. The variable **mshr83** is the magnitude which can be contoured.

Function List

- **max(grid1,grid2)**
Computes the maximum value at each gridpoint based on the values in the grids *grid1*, *grid2*, etc. More than two grids can be listed
- **min(grid1,grid2)**
Computes the minimum value at each gridpoint based on the values in the grids *grid1*, *grid2*, etc. More than two grids can be listed
- **sum(grid1,grid2)**
Computes the sum of each gridpoint based on the values in the grids *grid1*, *grid2*, etc. More than two grids can be listed.
- **add(grid1,grid2)**
Same as **sum**
- **avg(grid1,grid2)**
Computes the average at each gridpoint based on the values in the grids *grid1*, *grid2*, etc. More than two grids can be listed
- **interp(grid1,grid2|value,grid3)**
This interpolates between the grid specified in *grid1* and the grid in *grid3* based on the value specified either in *grid2* or the *value*. This does a simple linear interpolation. A value of **.5** would essentially give an average. A value of **.8** will return a value that is **.8** of the way to the second grid.
- **sub(grid1,grid2)**
This subtracts *grid2* from *grid1*.
- **thick(grid1,grid2)**
Same as **sub**
- **mul(grid1,grid2)**
Multiplies the grids in *grid1* and *grid2*.
- **div(grid1,grid2)**
Divides the values in *grid1* by the values in *grid2*.
- **mod(grid1,grid2)**
Does the modulus of *grid1* divided by *grid2*. This produces the remainder of the division computation.
- **sqrt(grid1)**
Does the square root of all values in *grid1*.
- **abs(grid1)**
Returns the absolute value of the values in *grid1*.
- **log(grid1)**
Takes the logarithm of each value in *grid1*
- **log10(grid1)**
Takes the logarithm base 10 of each value in *grid1*

- **exp(*grid1*)**
Exponentiates (e^x) the values in *grid1* . This is the inverse of the log function
- **pow(*grid1*,*grid2*)**
This performs for power function of the values in *grid1* raised to the power of the values in *grid2*.
- **sin(*grid1*)**
Performs the trigonometric function sine of the values in *grid1*. Input values are in degrees
- **cos(*grid1*)**
Performs the trigonometric function cosine of the values in *grid1*. Input values are in degrees
- **tan(*grid1*)**
Performs the trigonometric function tangent of the values in *grid1*. Input values are in degrees
- **asin(*grid1*)**
Performs the trigonometric function arcsine of the values in *grid1*. Output values are in degrees
- **acos(*grid1*)**
Performs the trigonometric function arccosine of the values in *grid1*. Output values are in degrees
- **atan(*grid1*)**
Performs the trigonometric function arctangent of the values in *grid1*. Output values are in degrees
- **atan2(*grid1*,*grid2*)**
Performs the trigonometric function arctangent of the values in *grid1* divided by *grid2*. The sign of 2 values determines the quadrant. This is the equivalent of the C atan2 function. Output values are in degrees
- **dewp(*grid1*,*grid2*)**
Computes the dewpoint based on temperature in *grid1* and the relative humidity in *grid2*
- **rhum(*grid1*,*grid2*)**
Computes the relative humidity based on temperature in *grid1* and the dewpoint in *grid2*
- **wetblb(*grid1*,*grid2*)**
Computes the wetbulb temperature based on temperature in *grid1* and the relative humidity or dewpoint in *grid2*.
- **wchill(*grid1*,*grid2*,[*grid3*])**
Computes the wind chill temperature based on the temperature in *grid1* and wind speed in *grid2*, or the U wind component in *grid2* and the V wind component in *grid3*.
- **heat(*grid1*,*grid2*)**
Computes the heat index temperature based on temperature in *grid1* and the relative humidity or dewpoint in *grid2*.
- **theta(*grid1*,[*grid2*])**
Computes the potential temperature based on temperature in *grid1* and the pressure in *grid2*. The pressure is optional if *grid1* is on a pressure surface (ie 500mb).
- **thetae(*grid1*,*grid2*,[*grid3*])**
Computes the equivalent potential temperature based on temperature in *grid1*, the relative humidity or dewpoint in *grid2* and the pressure in *grid3*. The pressure is optional if *grid1* is on a pressure surface (ie 500mb).
- **thetav(*grid1*,*grid2*,[*grid3*])**
Computes the virtual potential temperature based on temperature in *grid1*, the relative humidity or dewpoint in *grid2* and the pressure in *grid3*. The pressure is optional if *grid1* is on a pressure surface (ie 500mb).
- **vtemp(*grid1*,*grid2*,[*grid3*])**
Computes the virtual temperature based on temperature in *grid1*, the relative humidity or dewpoint in *grid2* and the pressure in *grid3*. The pressure is optional if *grid1* is on a pressure surface (ie 500mb).
- **vapor(*grid1*,[*grid2*])**
Computes the vapor pressure based on dewpoint in *grid1*, or the temperature in *grid1* and the relative humidity in *grid2*.
- **shum(*grid1*,[*grid2*],[*grid3*])**
Computes the specific humidity based on dewpoint in *grid1* and pressure in *grid2*, or temperature in *grid1*, relative humidity in *grid2* and pressure in *grid3*. The pressure is optional if *grid1* is on a pressure surface (ie 500mb).
- **mixrat(*grid1*,[*grid2*],[*grid3*])**
Computes the mixing ratio based on dewpoint in *grid1* and pressure in *grid2*, or temperature in *grid1*,

relative humidity in *grid2* and pressure in *grid3*. The pressure is optional if *grid1* is on a pressure surface (ie 500mb).

- **spd**(*grid1,grid2*)
Computes the wind speed based on U wind component in *grid1* and the V wind component in *grid2*.
- **mag**(*grid1,grid2*)
Computes the vector magnitude based on X component in *grid1* and the Y component in *grid2*.
- **dir**(*grid1,grid2*)
Computes the vector/wind direction based on X/U wind component in *grid1* and the Y V wind component in *grid2*.
- **ugeos**(*grid1*)
Computes the U geostrophic wind component based on geopotential height in *grid1*.
- **vgeos**(*grid1*)
Computes the V geostrophic wind component based on geopotential height in *grid1*.
- **uq**(*grid1,grid2*)
Computes the U Q-vector component based on geopotential height in *grid1* and temperature in *grid2*.
- **vq**(*grid1,grid2*)
Computes the V Q-vector component based on geopotential height in *grid1* and temperature in *grid2*.
- **dx**(*grid1*)
Finite differences ($\frac{dgrid1}{dx}$) the grid *grid1* in the X direction.
- **dy**(*grid1*)
Finite differences ($\frac{dgrid1}{dy}$) the grid *grid1* in the Y direction.
- **grad**(*grid*) => *xgrid ygrid*
Gradient of the grid. Outputs 2 grid vector, one for the X component and the second for the Y component.
- **vdiff**(*xgrid1,ygrid1,xgrid2,ygrid2*) => *xgrid ygrid*
Computes a vector difference. Outputs 2 grids: X and Y.
- **dot**(*xgrid1,ygrid1,xgrid2,ygrid2*)
Computes a dot product.
- **cross**(*xgrid1,ygrid1,xgrid2,ygrid2*) => *xgrid ygrid*
Computes a vector cross product. Outputs 2 grids: X and Y.
- **lapl**(*grid1*)
Computes the Laplacian of *grid1*.
- **conv**(*grid1,grid2,[grid3]*)
Computes the convergence of the vector/wind field based on the U wind component in *grid1* and the V wind component in *grid2*. If *grid3* is specified, the convergence is done on the grid field in *grid3*.
- **diverg**(*grid1,grid2,[grid3]*)
Computes the divergence of the vector/wind field based on the U wind component in *grid1* and the V wind component in *grid2*. If *grid3* is specified, the divergence is done on the grid field in *grid3*.
- **def1**(*grid1,grid2*)
Computes the first deformation term of the vector/wind field based on the U wind component in *grid1* and the V wind component in *grid2*.
- **def2**(*grid1,grid2*)
Computes the second deformation term of the vector/wind field based on the U wind component in *grid1* and the V wind component in *grid2*.
- **rvort**(*grid1,grid2*)
Computes the vorticity or curl of the vector/wind field based on the U wind component in *grid1* and the V wind component in *grid2*.
- **avort**(*grid1,grid2*)
Computes the absolute vorticity of the wind field based on the U wind component in *grid1* and the V wind component in *grid2*.
- **adv**(*grid1,grid2,grid3*)
Computes the advection of a specified field in *grid3* by the wind with the U wind component in *grid1* and the V wind component in *grid2*.

NOTE: The grid specifications can be replaced by numeric constants with the syntax:

#number

in most of the above functions. For example, the number 5.32 would be entered "#5.32".

Panel Plots

Overlay plots put one plot on top of another. The **panel** option allows plots to be separated into 4 or 6 panels. The attributes for the **panel** option are the panel size and location information:

- $nx \times ny [+x+y]$
These coordinates represent the relative panel coordinates. The **nx** and **ny** represent the number of plots in each direction. The **x** and **y** are the offsets measured from the upper left. For example, to produce a 4 panel plot (2x2) and the current plot is the lower left, the geometry would be:

2x2+0+1

- $dx \times dy [+x+y]$
These coordinates represent the fractional panel coordinates. The **dx** and **dy** represent the fraction of the overall window ($0 < dx < 1$) the panel will use. The **x** and **y** are the offsets measured from the lower left. For example, to produce a 4 panel plot (2x2) and the current plot is the upper left, the geometry would be:

.5x.5+0+.5

A sample panel setup:

```
c4p    Composite_4_panel    0 panel    [.5x.5+0+0],\
                        c850,\
                        panel    [.5x.5+.5+0],\
                        c700,\
                        panel    [.5x.5+0+.5],\
                        c500,\
                        panel    [.5x.5+.5+.5],\
                        c300
```

This will generate a 4 panel plot of composite plots. The upper left panel will be a composite 850 plot. The upper right panel will be a composite 700 mb plot. The lower left panel will be a composite 500 mb plot. The lower right panel will be a 300 mb composite plot.

SEE ALSO

Variable Lookup File (variable.lup)

The model name database file contains a list of model numbers from the GRIB files and a name to use in labels for WXP products. The model names change on a regular basis so rather than hard coding the model names into the program, the model lookup file provides the means for updating the model name list and to tailor how the model name is displayed on WXP products.

FORMAT

The model name file is formatted as follows:

```
num[num,...] abbr longname
num[num,...] abbr longname
...
```

where:

- *num* -- The variable/parameter ID from the GRIB product (Octet 9 of PDB). There can be more than one number associated with a variable. In some cases, there are special variables used with different models. For example, the Eta model uses 130 for sea level pressure whereas other models use 2 so pressure "**pres**" could be 4 different numbers so using the **pres** variable abbreviation can cover all possible sea level pressure variables.
- *abbr* -- The variable abbreviation to be used to specify the variable in WXP programs.
- *longname* -- The long name to use for labels using that variable. There can be spaces in the long name.

EXAMPLES

An example of a block from the **variable.lup** file:

```
1,2,129,130 pres Pressure
2 slpres Pressure
3 ptend Pressure tendency
4 pvort Potential vorticity
6 geop Geopotential
7 hght Geopotential Height
8 hght Height
10 ozone Total ozone
11 temp Temperature
12 vtemp Virtual temperature
13 theta Potential temperature
14 thetae Equiv potential temp
15 maxt Maximum temperature
16 mint Minimum temperature
17 dewp Dewpoint temperature
18 dewd Dewpoint depression
19 lapse Lapse rate
20 vis Visibility
24 plift Parcel lifted index
27 ganom Geopotential anomaly
31 wdir Wind direction
32 wspd Wind speed
32 spd Wind speed
33 uwnd U wind component
34 vwnd V wind component
35 strf Stream function
37 mstrf Montgomery stream func
38 svvel Vertical velocity-sigma
39 vvel Vertical velocity
```

40 gvvel Vertical velocity-geom
41 avort Abs vorticity
43 rvort Rel vorticity

...

SEE ALSO