NCAR Graphics Programmer Document

WMAP - A Package for Producing Daily Weather Maps and Plotting Station Model Data

Version 1.0

January, 1995

SCIENTIFIC COMPUTING DIVISION
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

Acknowledgments

Author: Fred Clare

Special thanks to Bob Lackman for reviewing this document. Also, thanks to Dennis Joseph and Ken Hansen for sharing their wind barb program.

Published by: National Center for Atmospheric Research, Scientific Computing Division, P.O. Box 3000, Boulder, CO 80307-3000. The National Center for Atmospheric Research is operated by the University Corporation for Atmospheric Research and is sponsored by the National Science Foundation. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Contents

WMAP, a Package for Plotting Daily Weather Maps and Station Model Data

1.0 INTRODUCTION

WMAP is a collection of subroutines that will allow the user to create daily weather maps like the ones appearing in major U.S. newspapers. Full color is available. Functionality also exists for plotting station model data (temperature, pressure, wind speed, cloud cover, and so forth) in accordance with the World Meteorological Organization (WMO) guidelines. It is also possible to plot icons representing daily weather (like “snow”, “thunderstorms,” and so forth).

The functionality of the package, and the documentation, is broken into: plotting front lines; plotting regions showing weather type or temperature zone; plotting city names and daily high/low temperature readings; plotting regional weather labels (like “BREEZY,” “HUMID,” and so forth); plotting regional temperature labels with optional arrows; plotting special symbols (like ones for high and low pressure) and icons representing daily weather; plotting map legends; plotting wind barbs and station model data.

Either C or Fortran procedures may be invoked to effect the above functionality. In the descriptions of the procedures, the Fortran entry will be described in detail. A synopsis of the equivalent C procedure follows.

The basic programming approach is to set values for internal parameters that control various aspects of the output and then invoke the appropriate C or Fortran procedure. For a complete list of, and an explanation of how to set and retrieve values for, the internal parameters available in this package, consult the section “Setting and Retrieving Internal Parameters” in Section 12. An easy way of using this package is to thumb through the example plots in Appendix A to find that plot that illustrates what you want to do, then modify the code that produces that plot to accomplish the desired goal. See Section 14.0 to find out how to retrieve and run the codes that produce the plots in Appendix A.

2.0 PLOTTING FRONT LINES

There is a single entry for drawing front lines.

Fortran:

```
CALL WMDRFT(N,X,Y)
```

where X and Y are linear arrays of dimension N and ((X(I),Y(I)),I=1,N) are world coordinate values.

C:

```
#include <ncarg/ncargC.h>

void c_wmdrft(int n, float *x, float *y)
```

where the parameters are analogous to the Fortran ones.

The coordinate values are used as control points for a spline curve fit. The curve passes through the given coordinates. If the front has little curvature, then only a few points need to be specified to get a nice smooth curve. See Figure 2 in Appendix A for an illustration.

2.1 Selecting front types.

The values of the internal parameters FRO and ALO specify which of the ten possible front types will be drawn by calling WMDRFT. FRO specifies the basic front type, and ALO indicates if the front is at the surface or aloft in the cases of warm, cold, or stationary fronts. See the table of internal parameters in Section 12.2 for details on the internal parameters FRO and ALO. See Figure 1 in Appendix A for a plot illustrating all front types.

2.2 The number of symbols appearing along a front line.

In the default case, WMAP attempts to choose a pleasing number of symbols along a front line by calculating the arc length of the line and spacing the symbols at equal distances along it. If the defaults are not desirable, complete control over the symbols appearing along the front line is possible as discussed in the next section.

2.3 Customizing the number and appearance of symbols on a front line.

Unless a value has been specified for the internal parameter NMS, which states explicitly how many symbols should appear along a front line (like the number of bumps on a warm front), the number of symbols is determined by the settings for the internal parameters BEG, , BET, and SWI. BEG specifies the distance along the front line before the first symbol is plotted; END specifies the distance along the front line after the final symbol is drawn; BET specifies the distance between symbols; SWI specifies the width of an individual symbol. The algorithm used is to find the largest number of symbols, NS, for which the sum of the four quantities BEG, END, NS*SWI, and (NS-1)*BET is less than the arc length of the front line. The distance between symbols is then readjusted to produce equal spacing. All distances are expressed as fractions of the maximum screen height. The value of the internal parameter ARC can be interrogated to determine the arc length of the most recent front line drawn.

If a non-negative value is specified for NMS, then precisely NMS symbols will be drawn along the front line, regardless of the settings for BEG, END, BET, and SWI. Setting the value of NMS to zero results in drawing a spline curve along the front line coordinates with no symbols appearing along it.

It is possible to specify the exact sequence in which warm front and cold front symbols appear along a front line, as well as which side of the line they point. The internal parameter array STY is used for this purpose. The legal values for elements of the STY array are plus or minus 1 and plus or minus 2. A “1” will produce a cold front symbol (a small triangle) and a “2” will produce a warm front symbol (a semi-circular bump). The algebraic sign of the elements of STY indicate what side of the front line the symbols will point - a positive value will cause the symbol to point to the right of the line (as traversed from the start to the end), and a negative value will cause the symbol to point to the left. Elements of the STY array are defined by using the internal parameter PAI to indicate which element is being specified and then providing a value for STY. The internal parameter REV can be used to reverse the directions that the symbols along a front line point.

The bumps that are drawn for warm front symbols are plotted as points along a Bezier curve. The default number of points along such a curve is 51 which makes these curves appear smooth at most sizes. If desired, this number can be reset by assigning the desired value to NBZ.

See the table of internal parameters in Section 12.2 for details on the internal parameters discussed in this section. See Figure 4 in Appendix A for an example that illustrates the use of the internal parameters BEG, BET, END, NMS, SWI, STY, PAI, and REV.

2.4 Setting colors.

The color index for warm front symbols (the semi-circular bumps) is specified by the value of WFC. In colored weather maps this symbol is usually drawn either in black or red; in black and white maps this symbol is black. The color index for cold front symbols (triangles) is specified by the value of CFC. In colored weather maps this symbol is usually drawn either in blue or black; in black and white maps this symbol is black. The alternating colors in a tropical front are specified by the values for the color indices assigned to the internal parameters T1C and T2C. In colored weather maps the colors for the alternating dashes in tropical fronts are usually red and green. The color for the front line itself is inherited from the color of the symbols along it. For fronts that do not have special symbols along them (convergence lines, instability lines, and intertropical fronts) the colors are specified by the value of the internal parameter COL.

See the table of internal parameters in Section 12.2 for details on the internal parameters discussed in this section. See Figures 1, 4, and 14 in Appendix A for examples that illustrate setting colors for front lines.

2.5 Endpoint slopes.

The value for the internal parameter SLF controls how slopes at the end points of a front line are to be handled. In the default case, the slopes are calculated internally as one-sided derivatives. SLF can be set to allow for user-specification of slopes at either end of a front line, or at both ends. The values of SL1 and SL2 specify the user-supplied slopes when applicable. The values of the internal parameters CS1 and CS2 can be interrogated to determine what values are currently being used for slopes at the endpoints.

See the table of internal parameters in Section 12.2 for details on the internal parameters discussed in this section. See Figure 3 in Appendix A for an example that illustrates setting endpoint slopes.

2.6 Linewidths.

The value of the internal parameter LIN specifies a linewidth scale factor for front lines (the thickness of the spline curve) for all fronts except the convergence lines, the instability lines, and the intertropical fronts. The value of the internal parameter DWD specifies the linewidth scale factor for the convergence lines, the instability lines, and the intertropical fronts. The internal parameter WTY is a flag indicating whether linewidths should be implemented via GKS linewidth calls, or whether linewidths should be simulated in software. If plotting to a device that does not have adjustable linewidths, then the latter should be chosen. Simulating linewidths in software is expensive.

See the table of internal parameters in Section 12.2 for details on the internal parameters LIN, DWD, and WTY.

3.0 PLOTTING REGIONS SHOWING WEATHER TYPE

The entry point for drawing regions indicating weather patterns or temperatures is:

Fortran:

```
CALL WMDRRG(N,X,Y,ITYPE,NC,XC,YC)
```

where X and Y are linear arrays of dimension N and ((X(I),Y(I)),I=1,N) are world coordinate values specifying a region. A spline fit is effected using the input coordinates so that only a few coordinates are needed to specify a smooth region. The spline fit passes through the input coordinates. See figure 2 in Appendix A for an example of a region defined by seven points. The region constructed is filled with either patterns or a solid color depending on ITYPE.

The legal values for ITYPE are:

ITYPE = ‘T’ for thunderstorms.

= ‘SH’ for showers.

= ‘R’ for rain.

= ‘F’ for snow flurries.

= ‘SN’ for snow.

= ‘I’ for ice.

= ‘INDEXnn’ where “nn” denotes a color index to use for the fill.

If NC is greater than 2, then the region resulting from the spline fit to ((X(I),Y(I)),I=1,N) will be clipped against the region in ((XC(I),YC(I)),I=1,NC). If no clipping is to be done, then set NC equal to 1.

C:

```
#include <ncarg/ncargC.h>

void c_wmdrrg(int n, float *x, float *y, char *itype, int nc, float *xc,
    float *yc)
```

where the parameters are analogous to the Fortran ones.

Internal parameter RHT can be used to change the size of the symbols in the weather patterns and COL can be used to change the colors. For examples of using WMDRRG, see Figures 2, 5, 13, 14 in Appendix A.

4.0 PLOTTING CITY NAMES AND DAILY HI/LOWs.

The entry point for plotting city names and daily HI/LOWs is:

Fortran:

```
CALL WMLABC(X,Y,CITY,TEMPS)
```

where X and Y are world coordinates specifying the position of the composite label that consists of the city name in the character variable CITY and the temperature ranges in the character variable TEMPS. For example, if

CITY='Boulder' and TEMPS='84/68', then these strings would be plotted with CITY on top and TEMPS below.

C:

```
#include <ncarg/ncargC.h>

void c_wmlabc(float x, float y, char *city, char *temps)
```

where the parameters are analogous to the Fortran ones.

There are four internal parameters that control the appearance of the city labels: CBC, CHT, CMG, and RFC. Each character in the city name and the HI/LOW value is drawn against a background that surrounds the character. The color of this background is controlled by the value of CBC and the extent of the background is controlled by the value of CMG. This background can be used to provide space around the labels for readability. The size of the labels is controlled by the value of CHT and the color of the characters is given by RFC.

Plotting dots to mark the city positions is facilitated by using the entry WMLABS which is described in Section 7.3 below.

For examples of plotting city names, hi/low data, and dots, see Figures 6, 13, and 14 in Appendix A.

5.0 PLOTTING REGIONAL WEATHER LABELS

The entry point for plotting regional weather labels (like "HOT", or "BREEZY", or "COLD") is:

Fortran:

```
CALL WMLABW(X,Y,LABEL)
```

where X and Y are world coordinates specifying the center position of the label in the character variable LABEL.

C:

```
#include <ncarg/ncargC.h>
```

```
void c_wmlabw(float x, float y, char *label)
```

where the parameters are analogous to the Fortran ones.

The labels are plotted in a rectangle that surrounds the label. This rectangle has a shadow. The internal parameters that control the appearance of the regional weather labels are: RC1, RC2, RC3, RC4, RC5, and WHT. WHT specifies the height of the characters in the label; RC4 specifies the color index of the text in the label; RC1 specifies a color index for the outline of the rectangle surrounding the text; RC2 specifies the color index of the interior of the box; RC3 specifies the color index of the rectangle shadow; and RC5 specifies the color index for outlines around each character in the LABEL (the character outlines are drawn only if RC5 is non-negative). For further details on these internal parameters, see Section 12.2 .

For examples of plotting regional weather labels, see Figures 6, 13, and 14 in Appendix A.

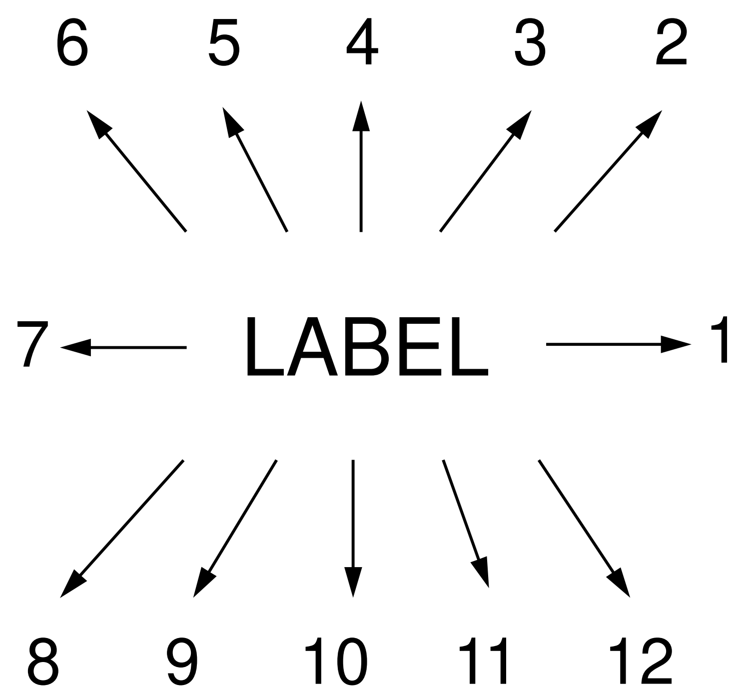
6.0 PLOTTING REGIONAL TEMPERATURE LABELS

The entry point for plotting regional temperature labels (like “90s”, or “60s”) is:

Fortran:

```
CALL WMLABT(X,Y,LABEL,IPOS)
```

where X and Y are world coordinates specifying a position for the label in the character variable LABEL. IPOS is a flag for controlling whether an optional arrow is drawn. If IPOS is 0, then no arrow is drawn and the label is centered at (X,Y). If IPOS is between 1 and 12, then arrows are drawn at the positions indicated in the following diagram:



If an arrow is drawn, then (X,Y) specifies the position at the tip of the arrow.

C:

```
#include <ncarg/ncargC.h>

void c_wmlabt(float x, float y, char *label, int ipos)
```

where the parameters are analogous to the Fortran ones.

The internal parameters that control the appearance of the regional temperature labels are: RBS, RFC, RLS, ROS, and THT. The appearance of the (optional) arrows is controlled by the internal parameters for arrows (see Section 7.2 for details on arrows). THT specifies the label height; RFC specifies the color index for the character color; RLS specifies the color index for character shadows; ROS specifies the color index for character outlines (drawn only if this index is non-negative). RBS specifies a color index for background boxes drawn around each character in the temperature label; these background boxes are drawn only if RBS is non-negative. RMG controls the size of the margins for the background boxes. For further details on these internal parameters, see Section 12.2 .

For examples of plotting regional temperature labels, see Figures 6, 13, and 14 in Appendix A.

7.0 PLOTTING SPECIAL SYMBOLS AND ICONS

The entry point for plotting special symbols and icons for daily weather is:

Fortran:

```
CALL WMLABS(X,Y,SYMTYP)
```

where X and Y are world coordinates specifying the center position of the symbol requested in the character variable SYMTYP. The available symbols include symbols for high and low pressure areas, arrows, dots, and icons representing daily weather (such as snow, rain, sun, etc.). These symbols are described in more detail in the subsections of this section.

C:

```
#include <ncarg/ncargC.h>

void c_wmlabs(float x, float y, char *symtyp)
```

where the parameters are analogous to the Fortran ones.

7.1 High and low pressure symbols.

If SYMTYP='HI', then a circular symbol with an “H” in it will be drawn at position (X,Y). The internal parameters controlling the appearance of the high pressure symbols are: SHT, HIF, HIB, HIS, and HIC. SHT controls the size of the symbol; HIF specifies the color of the “H” inside the circle; HIB specifies the color of the interior of the circle; HIS specifies the color of the symbol shadow; HIC specifies the color of the circumference of the circle. For further details on these internal parameters see Section 12.2 . For examples of the usage of high pressure symbols, see Figures 6, 13, 14 in Appendix A.

If SYMTYP='LOW', then a circular symbol with an "L" in it will be drawn at position (X,Y). The internal parameters controlling the appearance of the low pressure symbols are: SHT, LOF, LOB, LOS. SHT controls the size of the symbol; LOF specifies the color of the "L" inside the circle; LOB specifies the color of the interior of the circle; LOS specifies the color of the symbol shadow. For further details on these internal parameters see Section 12.2 . For examples of the usage of low pressure symbols, see Figures 6, 13, and 14 in Appendix A.

7.2 Arrows.

If SYMTYP='ARROW', then an arrow is drawn with the tip of the arrow at position (X,Y). The internal parameters that control the appearance of arrows are: ARS, AWC, ARD, ASC, AOC, ARL. ARS specifies the size of the arrow from tip to tail; AWC specifies the color of the arrow; ARD specifies the direction of the arrow in degrees counterclockwise from the positive X-axis; ASC specifies the shadow color (shadows are drawn only if ASC is non-negative); AOC specifies the arrow outline color (outlines are drawn only if AOC is non-negative); ARL specifies a scale factor for the arrow tails independent of the arrow size given by the value of ARS. Values of ARL greater than one will cause arrows to have longer tails, and values less than one will cause shorter tails. For further details on these internal parameters see Section 12.2 . For examples of the usage of arrows, see Figures 6, 13, and 14 in Appendix A.

7.3 Dots.

If SYMTYP='DOT', then a dot is drawn with its center at coordinate (X,Y). The internal parameters that control the appearance of dots are: DBC, DTC, and DTS. DTS specifies the size of the dot; DTC specifies the color index of the center of the dot; DBC specifies the color of the background circle of the dot. For further details on these internal parameters see Section 12.2 . For examples of the usage of dots, see Figures 6, 13, and 14 in Appendix A.

7.4 Icons for daily weather.

Icons for daily weather can be drawn for many weather conditions by setting SYMTYP to one of the following:

1 'CLOUD'

2 'ICE'

3 'IS' (Intermittent showers)

4 'IT' (Sunny, chance of T-storms)

5 'MC' (Mostly cloudy)

6 'MS' (Mostly sunny)

7 'RAIN'

8 'RS' (Rain and snow)

9 'SNOWFLAKES'

10 'SUN'

11 'THUNDERSTORM'

12 'WIND'

Two internal parameters that apply to all of the weather icons are SHT (specifies height) and COL (specifies color for single objects). There are three symbols in the icons that have special controls: clouds, lightening bolts, and suns:

7.4.1 clouds

There are three internal parameters that control the appearance of cloud symbols that appear in the daily weather icons:CC1, CC2, and CC3. CC1 specifies the color of the interior of the cloud; CC2 specifies the color of the cloud shadow; CC3 specifies the color of the outline of the cloud. For further details on these internal parameters see Section 12.2 . For examples of the usage of cloud symbols, see Figures 7 and 8 in Appendix A.

7.4.2 lightening bolts

There are three internal parameters that control the appearance of lightening bolt symbols that appear in the daily weather icons: LC1, LC2, and LC3. LC1 specifies the color of the interior of the lightening bolt; LC2 specifies the color of the Lightening bolt shadow; LC3 specifies the color of the outline of the lightening bolt. For further details on these internal parameters see Section 12.2 . For an example of the usage of the lightening bolt symbol, see Figures 7 in Appendix A.

7.4.3 sun

There are four internal parameters that control the appearance of the sun symbols that appear in the daily weather icons: SC1, SC2, SC3, and SC4. SC1 specifies the color of the center of the sun; SC2 specifies the color of the points of the sun; SC3 specifies the color of the outline of the sun; SC4 specifies the color of the sun shadow. For further details on these internal parameters see Section 12.2 . For examples of the usage of sun symbols, see Figures 7 and 8 in Appendix A.

8.0 PLOTTING MAP LEGENDS

The entry point for plotting legends at the bottom of a weather map is:

Fortran:

```
CALL WMLGND(X,Y,NTYPE,IROWS,ICOLS)
```

where X and Y are world coordinates. NTYPE specifies what legends are to be drawn:

NTYPE	=	1	Plot the legend for the weather types. There are six weather types: showers; T-storms; rain; flurries; snow; ice. The coordinate (X,Y) for this type specifies the lower left corner of the legend.
	=	2	Plot the legend for front types. There are three front types labeled: cold; warm; stationary. The (X,Y) coordinate for this legend specifies the lower right corner of the legend.
	=	3	Plot the explanatory legend. The (X,Y) coordinate for this legend specifies the bottom center of the legend.

IROWS and ICOLS specify how many rows and columns there will be in displaying the weather types; these values are significant only when NTYPE=1. Choices for IROWS x ICOLS are: 1x6, 2x3, 3x2, and 6x1.

C:

```
#include <ncarg/ncargC.h>

void c_wmlabs(float x, float y, int ntype, int irows, int icols)
```

where the parameters are analogous to the Fortran ones.

The internal parameters that apply to the appearance of legends are COL for specifying color and those parameters that apply to the appearance of weather fronts. See Section 2.0 for details on the internal parameters that apply to weather fronts. For examples of weather map legends, see Figures 13 and 14 in Appendix A.

9.0 WIND BARBS AND STATION MODEL DATA

9.1 Wind Barbs

The entry point for plotting wind barbs is:

Fortran:

```
CALL WMBARB(X,Y,U,V)
```

where X and Y are world coordinates specifying the center position of the base of the barb and U and V are the components of the wind vector. The angle at which the wind barb is drawn is determined by the vector (U,V) and the magnitude of that vector denotes the wind speed in knots, plotted as per the chart in Figure 9 of Appendix A.

C:

```
#include <ncarg/ncargC.h>

void c_wmbarb(float x, float y, float u, float v)
```

where the parameters are analogous to the Fortran ones.

The internal parameters that apply to the appearance of wind barbs are COL, WBA, WBD, WBF, WBS, WBT.

COL specifies the color of the barb; WBA specifies the angle, in degrees, that the wind barb tick marks make with the wind barb shaft; WBD specifies the spacing between tick marks along the shaft, expressed as a fraction of the shaft length; WBF is a flag indicating whether space should be left at the base of a wind barb for a sky cover circle; WBS specifies the size of the wind barb shaft; WBT specifies the length of a wind barb full tick mark. See Section 12.2 for details on the internal parameters that apply to wind barbs. For examples of wind barb usage, see Figures 10, 11, and 12 in Appendix A.

9.2 Plotting Station Model Data

The entry point for plotting station model data is:

Fortran:

```
CALL WMSTNM(X,Y,IMDAT)
```

where X and Y are world coordinates specifying the center position of the base of the wind barb shaft. IMDAT is a CHARACTER*5 array of dimension 10 encoded as per the WMO/NOAA guidelines. In more detail:

IMDAT(1)(1:1)	=	i_R	-	the precipitation data indicator
IMDAT(1)(2:2)	=	i_X	-	weather data and station type indicator
IMDAT(1)(3:3)	=	h	-	height above ground of base of lowest cloud
IMDAT(1)(4:5)	=	VV	-	visibility in miles and fractions
IMDAT(2)(1:1)	=	N	-	total amount of cloud cover
IMDAT(2)(2:3)	=	dd	-	direction from which wind is blowing
IMDAT(2)(4:5)	=	ff	-	wind speed in knots

For I=3,9 if IMDAT(I)(1:1) =

‘1’,	then	IMDAT(I)(2:2)	=	s_n	-	sign of temperature
		IMDAT(I)(3:5)	=	TTT	-	current air temperature
‘2’,	then	IMDAT(I)(2:2)	=	s_n	-	sign of dew point temperature
		IMDAT(I)(3:5)	=	T_d	-	dew point
‘3‘,	then	IMDAT(I)(2:5)	=	PO	-	station pressure (not plotted)
‘4’,	then	IMDAT(I)(2:5)	=	PPP P	-	pressure reduced to sea level
‘5’,	then	IMDAT(I)(2:2)	=	a	-	characteristic of barograph
		IMDAT(I)(3:5)	=	ppp	-	pressure change, last 3 hrs.
‘6’,	then	IMDAT(I)(2:4)	=	RRR	-	precipitation
		IMDAT(I)(5,5)	=	t_R	-	time duration of precipitation
‘7’,	then	IMDAT(I)(2:3)	=	ww	-	present weather
		IMDAT(I)(4:4)	=	W_1	-	most significant past weather

	IMDAT(I)(5:5)	=	W_2	-	2nd most sig. past weather
'8', then	IMDAT(I)(2:2)	=	N_h	-	Fraction of sky cover
	IMDAT(I)(3:3)	=	C_L	-	cloud type, low clouds
	IMDAT(I)(4:4)	=	C_M	-	cloud type, medium clouds
	IMDAT(I)(5:5)	=	C_H	-	cloud type, high clouds

C:

```
#include <ncarg/ncargC.h>

void c_wmstnm(float x, float y, char *imdat)
```

where the parameters are analogous to the Fortran ones.

The internal parameters that control the appearance of wind barbs apply to the wind barbs that are part of the station model display. See Section 9.1 for details on wind barbs. In addition to the wind barb parameters, the parameters WBC and WBL apply to the station model display. WBC specifies the diameter of the sky cover circle at the base of the wind barb and WBL specifies the size of the text labels in the station model display.

See Section 12.2 for details on the internal parameters that apply to the display of station models and wind barbs. For examples of station mode displays, Figures 11 and 12 in Appendix A.

10.0 SETTING DEFAULTS

There is a single call that returns all settings of all internal parameters to their default values:

Fortran:

```
CALL WMDFLT()
```

C:

```
#include <ncarg/ncargC.h>
```

```
void c_wmdflt()
```

11.0 PUTTING IT ALL TOGETHER

The plots in Figures 13 and 14 utilize almost all of the functionality discussed in this document to produce sample weather maps for the continental U.S. Figure 13 is a grayscale map and figure 14 is a full color map. Of course any other part of the world could have been used equally as well.

12.0 INTERNAL PARAMETERS

12.1 Setting and retrieving values for internal parameters

As mentioned in the introduction, the basic functionality of the package is implemented by way of setting values for internal parameters to control various attributes and then invoking one of the basic functions. The three functions for setting values for internal parameter and the three functions for retrieving internal parameter values are described in this section.

12.1.1 Settingnteger values.

Fortran:

```
CALL WMSETI(CNP, IVP)
```

where CNP is a character string at least three characters long (whose legal values are those given in the table of internal parameters), and IVP is the integer value to be assigned to that parameter.

C:

```
#include <ncarg/ncargC.h>

void c_wmseti(char *cnp, int ivp)
```

where the parameters are analogous to the Fortran ones.

12.1.2 Setting real values.

Fortran:

```
CALL WMSETR(CNP, RVP)
```

where CNP is a character string at least three characters long (whose legal values are those given in the table of internal parameters), and RVP is the real value to be assigned to that parameter.

C:

```
#include <ncarg/ncargC.h>

void c_wmsetr(char *cnp, float rvp)
```

where the parameters are analogous to the Fortran ones.

12.1.3 Setting string values.

Fortran:

```
CALL WMSETC(CNP, CVP)
```

where CNP is a character string at least three characters long (whose legal values are those given in the table of internal parameters), and CVP is the character value to be assigned to that parameter.

C:

```
#include <ncarg/ncargC.h>

void c_wmsetc(char *cnp, char *cvp)
```

where the parameters are analogous to the Fortran ones.

12.1.4 Retrieving integer values.

Fortran:

```
CALL WMGETI(CNP, IVP)
```

where CNP is a character string at least three characters long (whose legal values are those given in the table of internal parameters), and IVP is returned as the current integer value assigned to that parameter.

C:

```
#include <ncarg/ncargC.h>

void c_wmgeti(char *cnp, int ivp)
```

where the parameters are analogous to the Fortran ones.

12.1.5 Retrieving real values.

Fortran:

```
CALL WMGETR(CNP, RVP)
```

where CNP is a character string at least three characters long (whose legal values are those given in the table of internal parameters), and RVP is returned as the real value currently assigned to that parameter.

C:

```
#include <ncarg/ncargC.h>

void c_wmgetr(char *cnp, float rvp)
```

where the parameters are analogous to the Fortran ones.

12.1.6 Retrieving string values.

Fortran:

```
CALL WMGETC(CNP, CVP)
```

where CNP is a character string at least three characters long (whose legal values are those given in the table of internal parameters), and CVP is the character value currently assigned to that parameter. CVP should be of a length suitable to receive the string value returned.

C:

```
#include <ncarg/ncargC.h>

void c_wmgetc(char *cnp, char *cvp)
```

where the parameters are analogous to the Fortran ones.

12.2 Table of internal parameters

The internal parameters of WMAP are summarized in the following table.

Name	Type	Description	Default
ALO	integer	Flag to indicate whether a weather front is at the surface or aloft (ALO=0 is surface; ALO=1 is aloft).	0
AOC	integer	Color index for the outlines of arrows (outlines drawn only if AOC is non-negative).	-1
ARC	float	Length of current front line (retrieval only)	0.
ARD	float	Direction of arrows, expressed in degrees.	0.
ARL	float	Scale factor for the length of an arrow's tail, independent of the arrow's size.	1.
ARS	float	Size of arrows, expressed as a fraction of the maximum screen height.	0.035
ASC	integer	Color index for the shadows of arrows (shadows are drawn only if ASC is non-negative).	-1
AWC	integer	Color index for the interior of arrows.	1
BEG	float	Space, expressed in world coordinates, to leave along a front line before the first symbol is drawn.	0.015
BET	float	Space, expressed in world coordinates, to leave between symbols along a front line.	0.045
CBC	integer	The color index to be used for backgrounds of city and daily high/low labels.	1
CC1	integer	Color index for the interior of a cloud symbol.	2
CC2	integer	Color index for the outline of the cloud symbol.	1
CC3	integer	Color index for the shadow of the cloud symbol.	1
CFC	integer	Color index to use for cold front symbols.	1

Name	Type	Description	Default
CHT	float	Height of characters, expressed as a fraction of the maximum screen width, of the city labels and daily high/low temperatures.	0.0105
CMG	float	Size, expressed as a fraction of the maximum screen height, of the margins used for the city labels.	0.002
COL	integer	Color index to use for all objects that require only a single color setting.	1
CS1	float	Slope of the left edge of a front line as calculated internally as measured in degrees from the X-axis (used when SLF=0,2, or 3). This parameter is for retrieval only.	N/A
CS2	float	Slope of the right edge of a front line as calculated internally as measured in degrees from the X-axis (used when SLF=0,1, or3). This parameter is for retrieval only.	N/A
DBC	integer	The color index to be used for the background shadow for the dots marking the city locations.	0
DTC	integer	The color index to use for the dots marking the city locations.	1
DTS	float	Size, expressed as a fraction of the maximum screen height, of the dots used to mark cities.	0.006
DWD	float	Line widths for front lines that do not have symbols along them (like tropical fronts and convergence lines).	2.0
END	float	Space, expressed in world coordinates, to leave along a front line after the last symbol has been drawn.	0.015
FRO	string	Front type (one of 'WARM', 'COLD', 'OCCLUDED', 'STATIONARY', 'SQUALL', 'TROPICAL', or 'CONVERGENCE').	'WARM'
HIB	integer	Background color index for the "H" drawn for the high pressure symbols.	0
HIC	integer	Color index of the circumscribed circle for the "H" drawn for the high pressure symbols.	1
HIF	integer	Color index for the "H" drawn for the high pressure symbols.	1
HIS	integer	Color index for the shadow of the high pressure symbols	1
LC1	integer	Color index for the interior of the lightening bolt symbol	2
LC2	integer	Color index for the outline of the lightening bolt symbol.	1
LC3	integer	Color index for the shadow of the lightening bolt symbol	1
LIN	float	Line widths, expressed as a fraction of the maximum screen width, for fronts having symbols along them.	8.

Name	Type	Description	Default
LOB	integer	Color index for the background of the "L" drawn for the low pressure symbols.	1
LOF	integer	Color index for the "L" drawn for the low pressure symbols.	0
LOS	integer	Color index for the shadow of the low pressure symbols.	0
LWD	float	Line width used when parameter WTY=1 (see below)	0.00275
MXS	integer	Maximum number of symbols allowed along a weather front line.	200
NBZ	integer	The number of points to use in the Bezier curves for the symbols along the warm fronts.	51
NMS	integer	Specifies precisely the number of symbols to appear along a weather front line (if this parameter has not been set by the user, then it is calculated internally).	-1
PAI	integer	Current parameter array index used in specifying internal parameters that are arrays.	1
RBS	integer	The color index to use for the background of the regional temperature labels (plotted only if RBS is non-negative).	-1
RC1	integer	Color index for the outline of the boxes drawn for the regional weather labels.	1
RC2	integer	Color index for the backgrounds of the boxes used for the regional weather labels.	0
RC3	integer	Color index for the shadow color of the boxes used for the regional weather labels.	1
RC4	integer	Color index for the text string used for regional weather labels.	1
RC5	integer	Color for the outlines of the characters in the text strings used for regional weather labels (plotted only if RC5 is non-negative).	-1
REV	integer	Reverses the current setting for the direction symbols will be drawn along front lines.	N/A
RFC	integer	The color index to be used for the foreground of regional temperature labels and cities.	1
RHT	float	Height, expressed as a fraction of the maximum screen width, of the characters used for the regional weather patterns (like rain, snow, etc.).	0.008
RLS	integer	The color index to use for shadows of regional temperature labels.	1
RMG	float	Size, expressed as a fraction of the maximum screen height, of the margins of the regional temperature labels.	0.001

Name	Type	Description	Default
ROS	integer	The color index to use for the outlines of the regional temperature labels (plotted only if ROS is non-negative).	-1
SC1	integer	Color index to be used for the center of the sun symbol.	2
SC2	integer	Color index for the points of the sun symbol.	3
SC3	integer	Color index for the outline of the sun symbol.	1
SC4	integer	Color index for the shadow of the sun symbol.	1
SHT	float	Height of symbols, expressed as a fraction of the maximum screen width, for all the special symbols.	0.02
SL1	float	The slope of the beginning of a front line measured in degrees from the X-axis. This parameter is used in conjunction with the parameter SLF.	0.0
SL2	float	The slope of the end of a front line measured in degrees from the X-axis. This parameter is used in conjunction with the parameter SLF.	0.0
SLF	integer	Flag for indicating how the slopes at the end of a front line should be handled (0=use SL1 & SL2; 1=use SL1 only; 2=use SL2 only; 3=use neither SL1 or SL2). When either SL1 or SL2 is not used, it is calculated internally.	3
STY	integer	An array for precisely specifying whether a warm front or cold front symbol is to be drawn at the specified position along a front line (1=cold; 2=warm). Use the internal parameter PAI for defining this array.	all 2s
SWI	float	Width of a symbol along a weather front, expressed as a fraction of the maximum screen width.	0.0325
T1C	integer	One color to use for the alternating colors of the dashes in the tropical fronts.	1
T2C	integer	A second color to use for the alternating colors of the dashes in the tropical fronts.	1
THT	float	Height of characters, expressed as a fraction of the maximum screen width, for the regional temperature labels.	0.0165
WBA	float	Angle (in degrees) that the wind barb tics make with the wind barb shafts.	62.
WBC	float	Diameter of sky cover circle at base of wind barb, expressed as a fraction of the shaft length.	0.3
WBD	float	Spacing between tick marks along a wind barb expressed as a fraction of the wind barb shaft length.	0.1
WBF	integer	Flag indicating whether the base of a wind barb should be drawn to allow for the sky cover circle at its base (WBF=1 means yes; WBF=0 means no).	0

Name	Type	Description	Default
WBL	float	Size of the text labels in the station model display, expressed as a fraction of the shaft length.	0.17
WBR	float	Radius of the larger circle drawn for calm, as a fraction of the wind barb shaft length.	0.25
WBS	float	The size, expressed as a fraction of the maximum screen height, of a wind barb shaft.	0.035
WBT	float	Length of wind barb full tic as a fraction of its shaft length.	0.33
WFC	integer	Color index to use for warm front symbols.	1
WHT	float	Height of characters, expressed as a fraction of the maximum screen width, of the characters used for regional weather labels (plotted with WMLABW or c_wmlabw).	0.014
WTY	integer	Flag indicating whether linewidths are to be implemented via GKS (WTY=0), or simulated internally (WTY=1).	0

13.0 ERROR HANDLING

When a WMAP routine detects an error condition, it calls the routine SETER, which is the principal routine in the error-handling package for NCAR Graphics. (There is a programmer document describing SETER and associated routines; see that document for complete information about error handling in NCAR Graphics.)

By default, SETER prints a line and STOPs. The line printed will look something like this:

```
ERROR    2 IN WMDRFT - not enough space along the
input curve to draw two symbols for a stationary front
```

The error number (“2”, in the example) may be of use to a consultant (to determine exactly where the error occurred), but is not otherwise meaningful. The actual error message consists of the name of the routine in which the error occurred (“WMDRFT”, in the example), a blank, a minus sign, another blank, and, lastly, a short description of the error.

All errors are "recoverable" in the sense that, if the user program puts SETER into "recovery mode", control will be returned to the caller of the WMAP routine in which the error occurred. In some cases, it is then possible to take remedial action to get around whatever problem has occurred; in any case, the error flag can be cleared and execution of the user's program can continue.

14.0 EXAMPLES

On a Unix system on which NCAR Graphics has been installed, the command "ncargex" may be used to acquire the code for and create all of the example plots contained in Appendix A of this document. The example names recognized by ncargex are of the form "wmex*nn*" where "*nn*" is the Figure number used in Appendix A.

Appendix A

Example Plots

Figure 1 - ncargex example wmex01

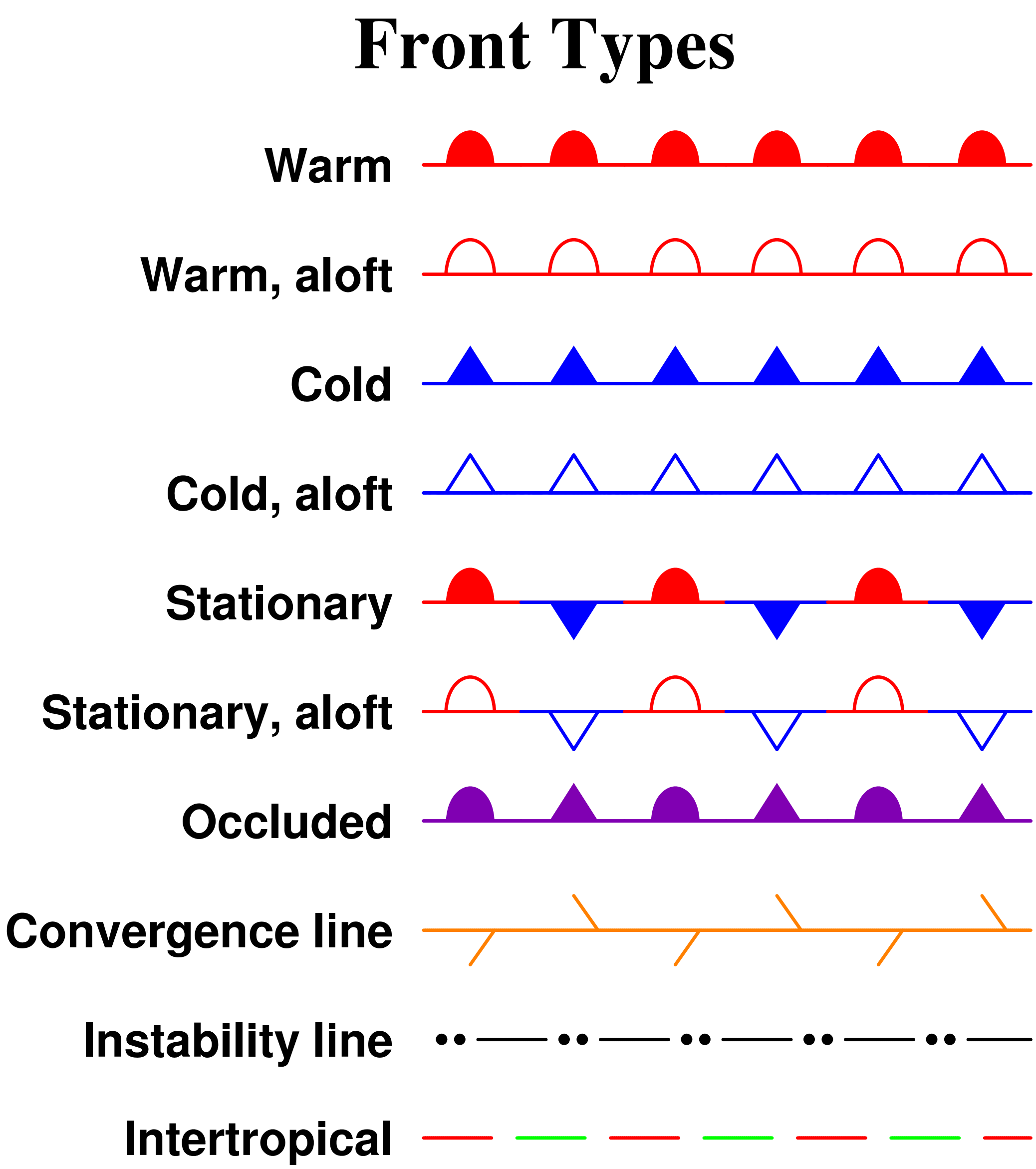


Figure 2 - ncargex example wmex02

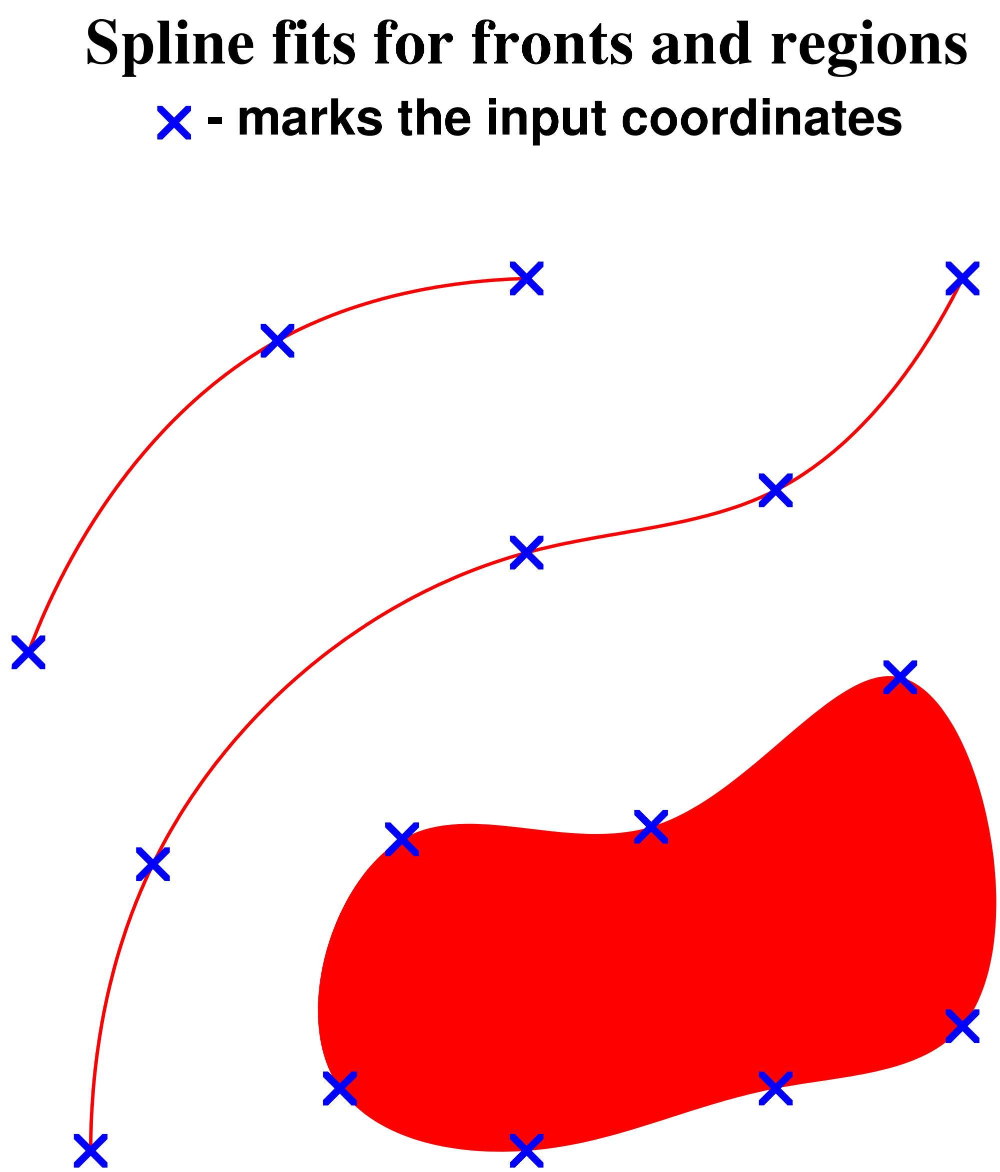


Figure 3 - ncargex example wmex03

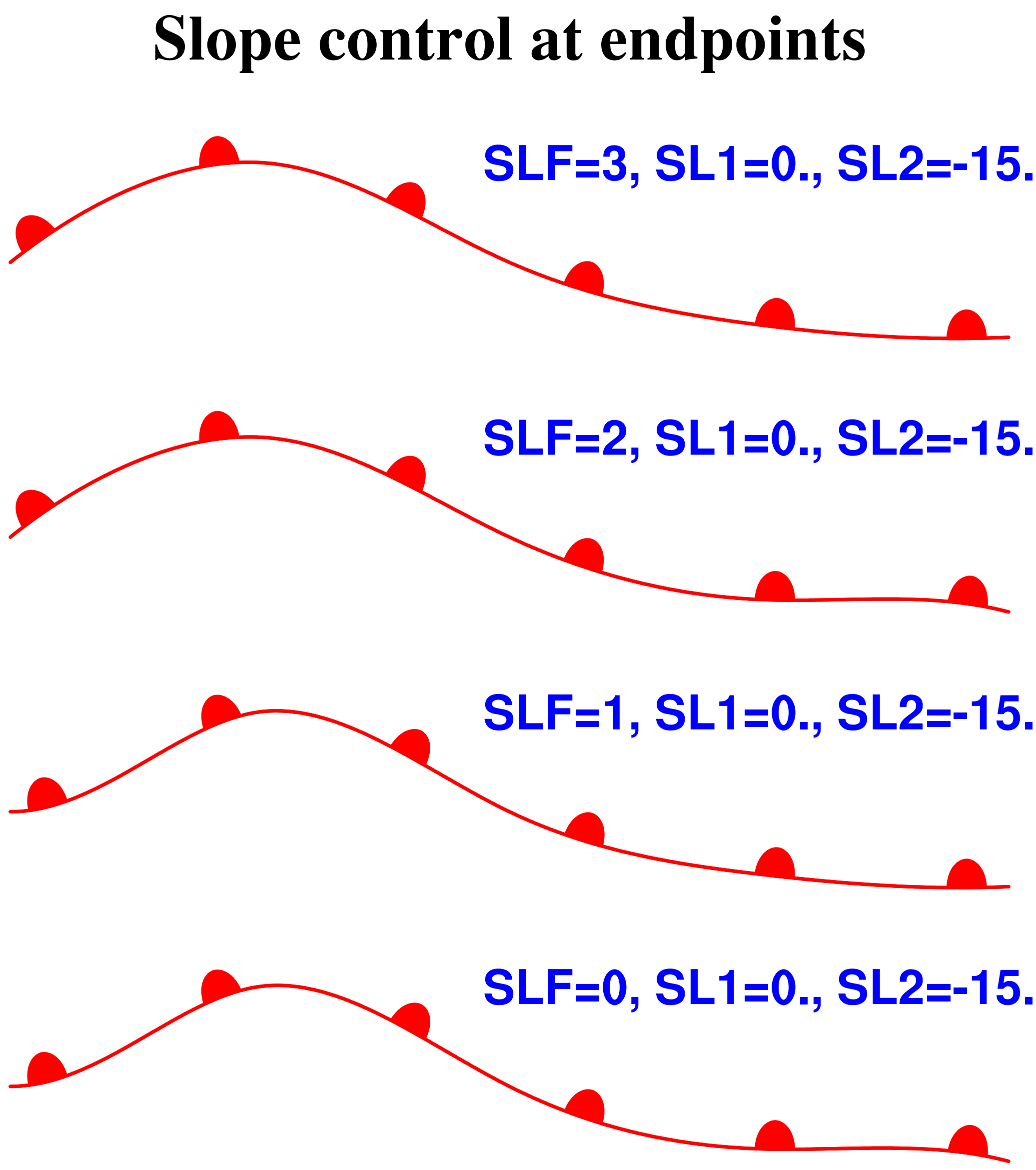


Figure 4 - ncargex example wmex04

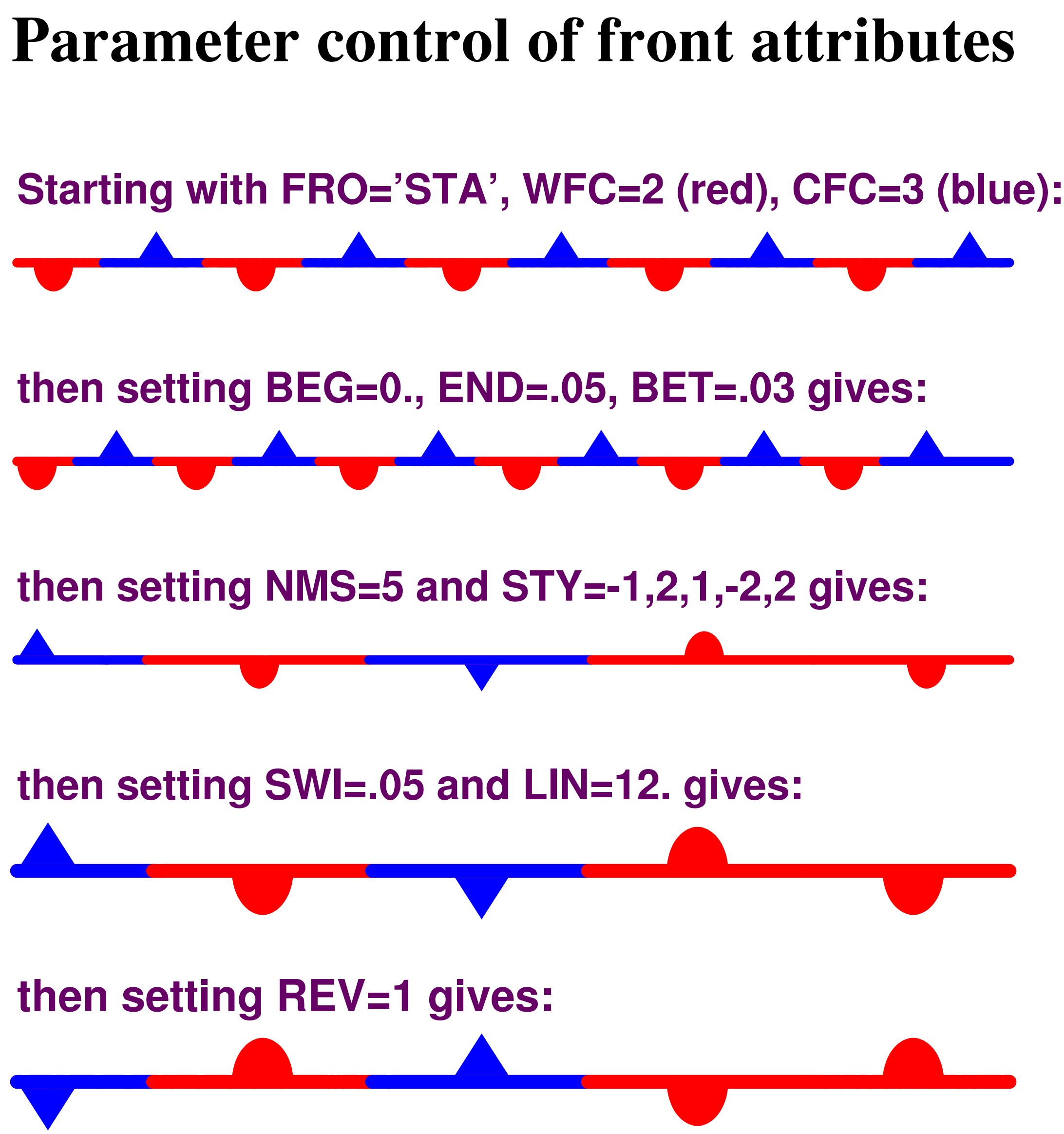


Figure 5 - ncargex example wmex05

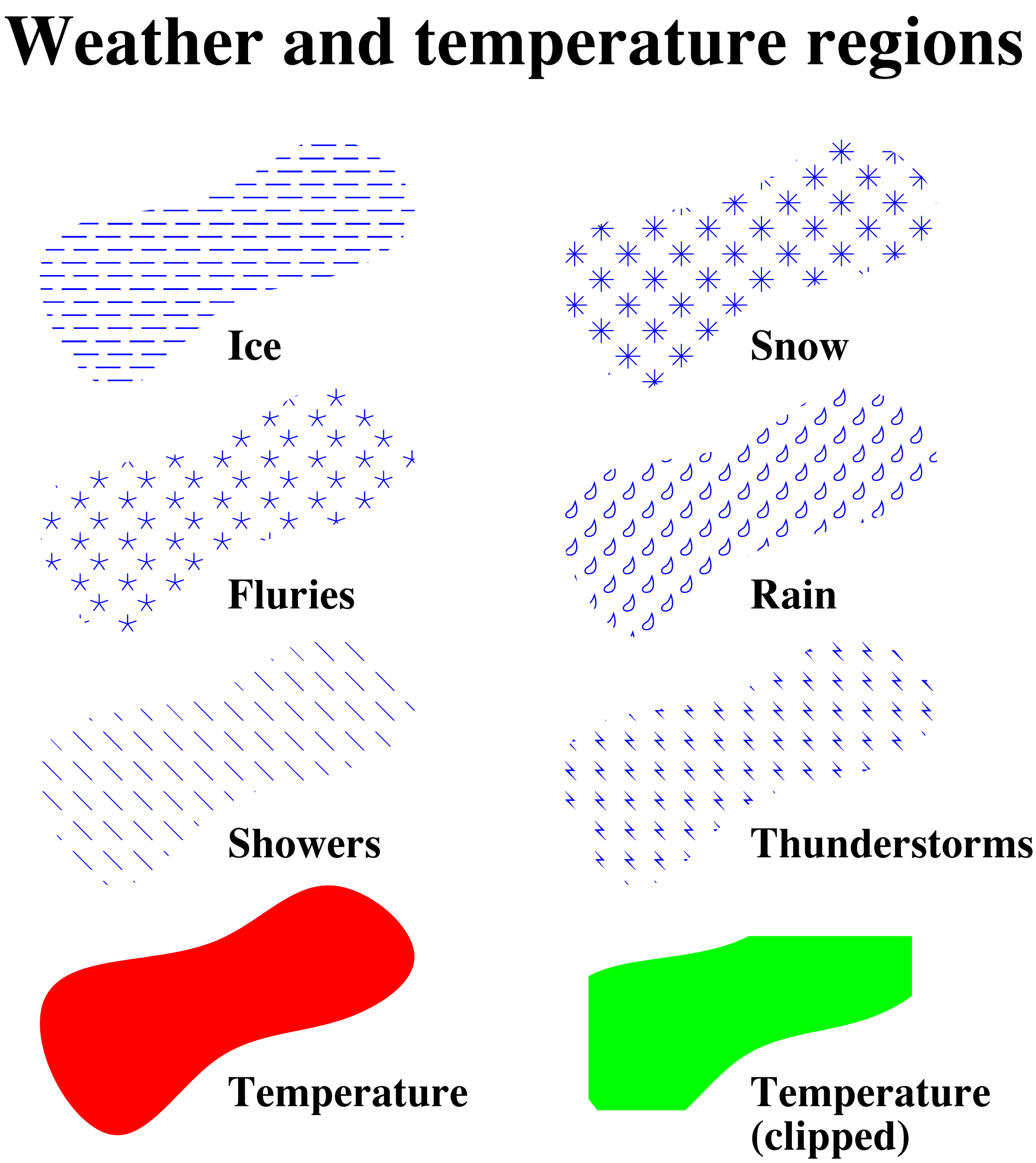


Figure 6 - ncargex example wmex06

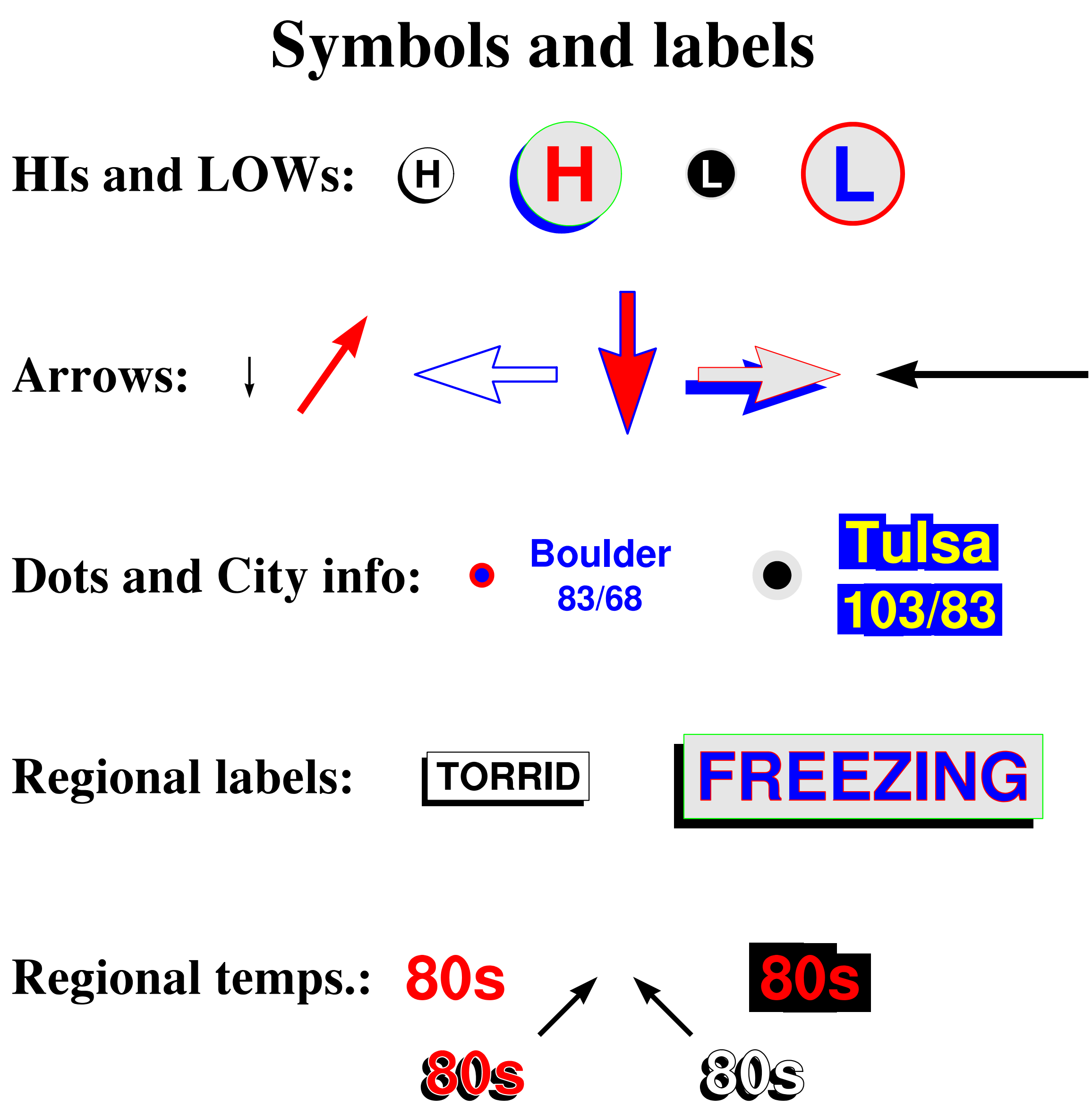


Figure 7 - ncargex example wmex07

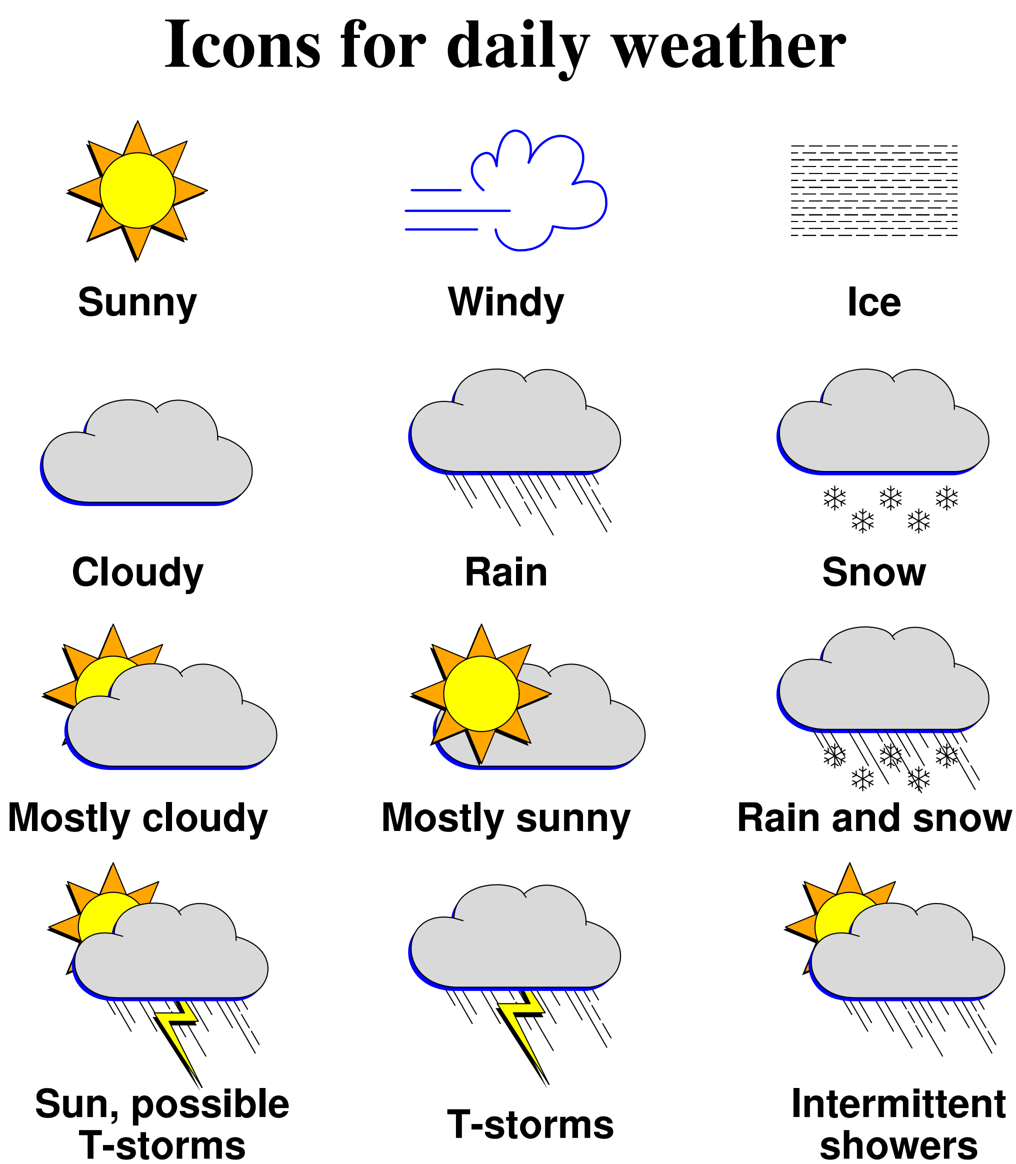


Figure 8 - ncargex example wmex08

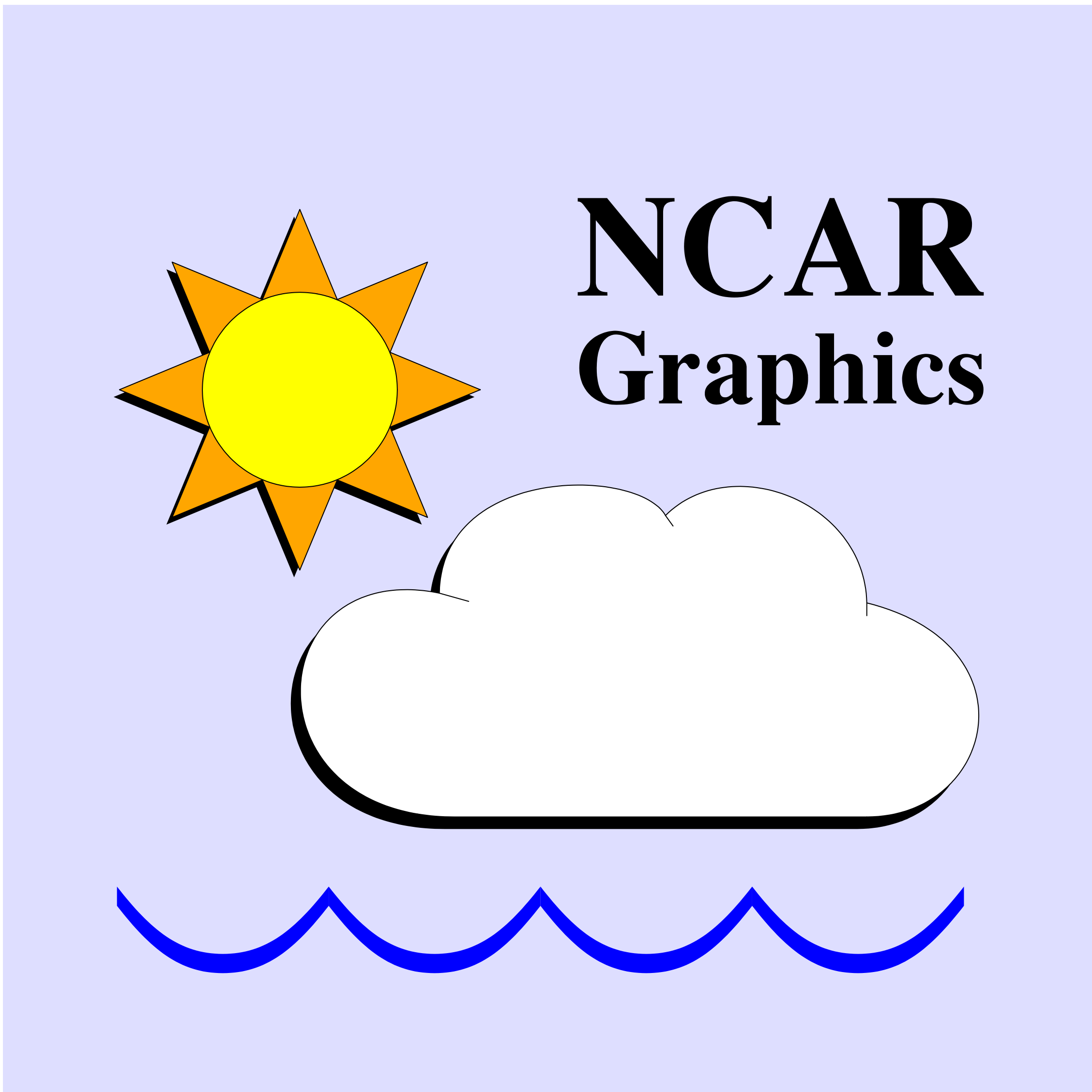


Figure 9 -ncargex example wmex09

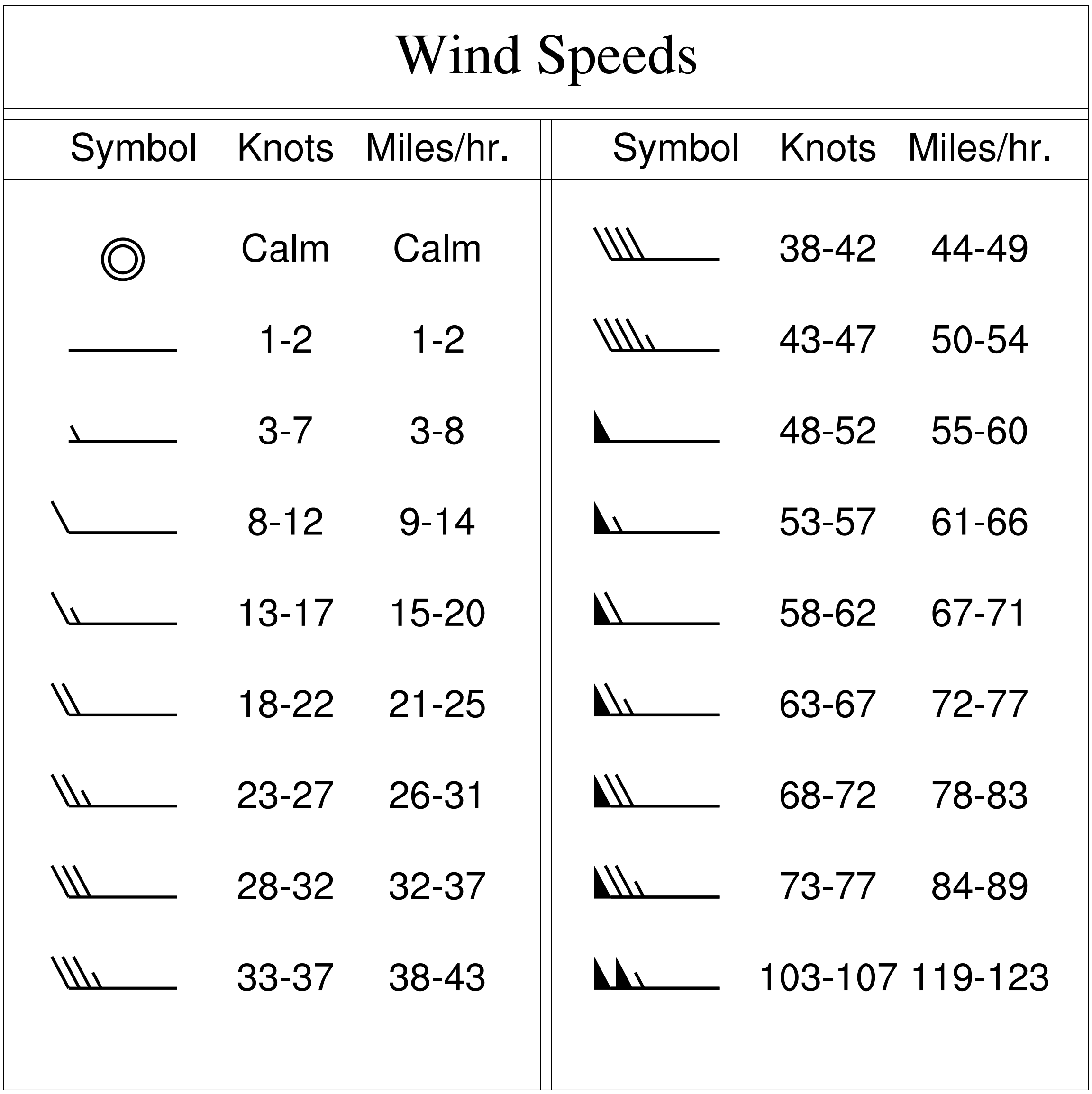


Figure 10 - ncargex example wmex10

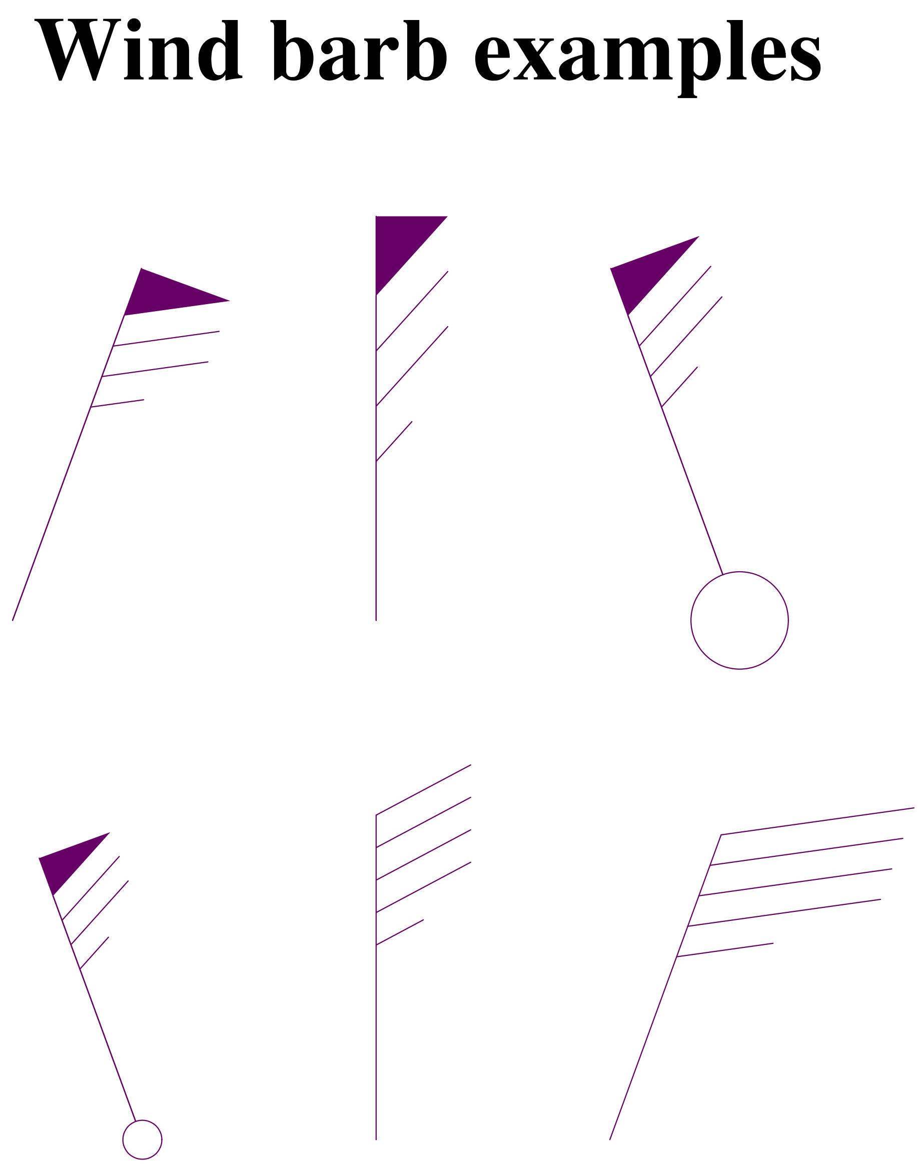


Figure 11 - ncargex example wmex11

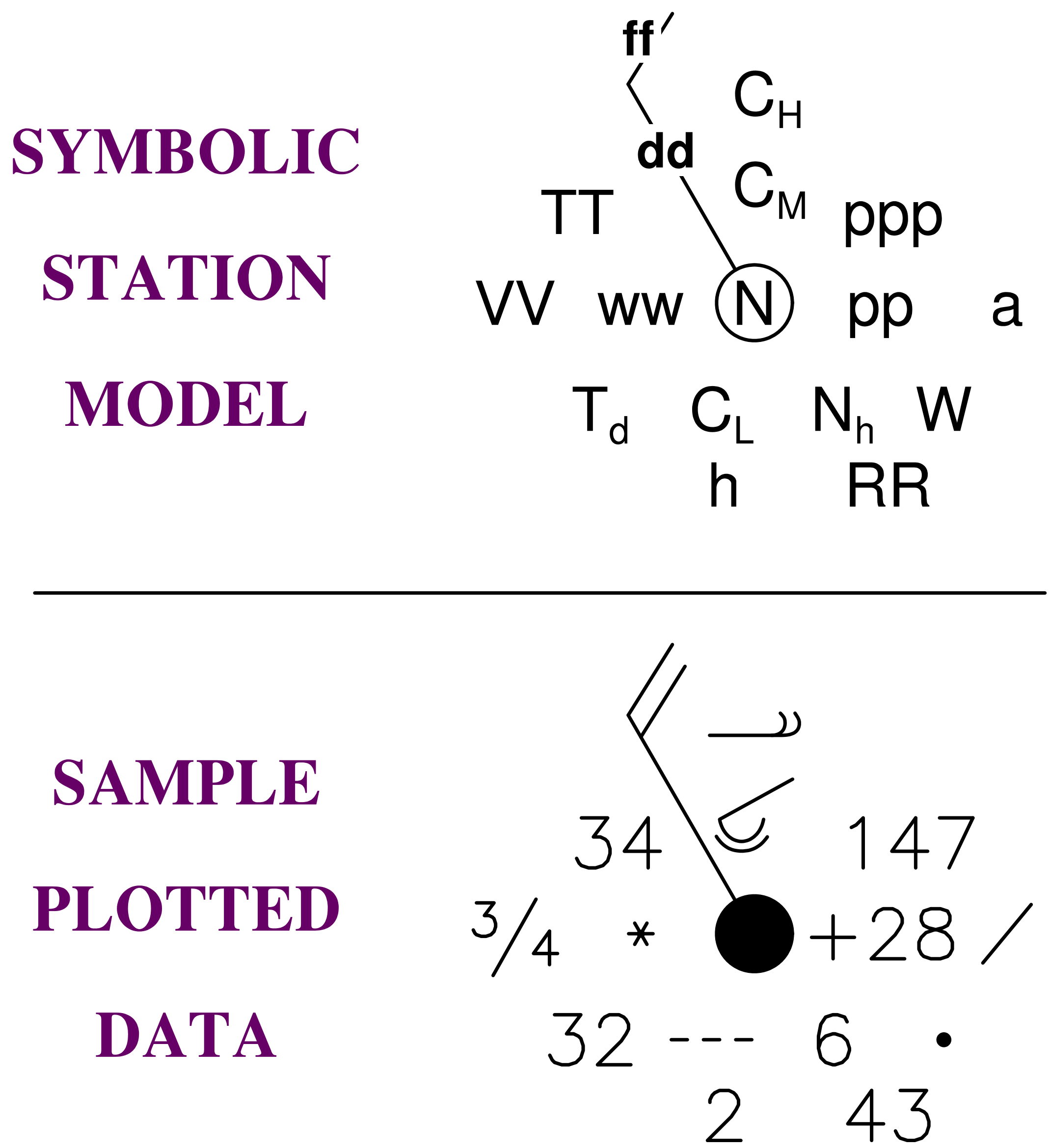


Figure 12 - ncargex example wmex12

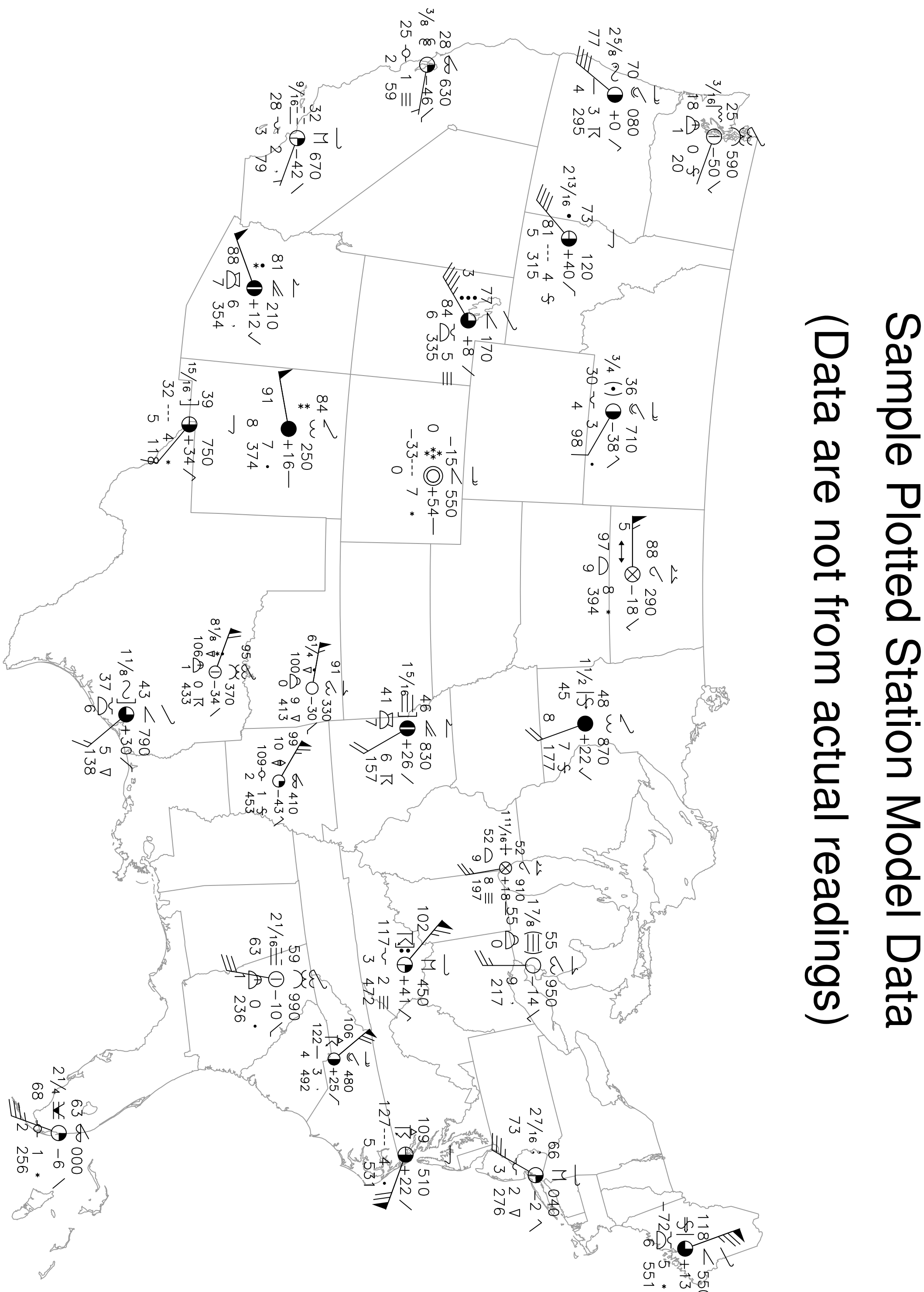


Figure 13 - ncargex example wmex13

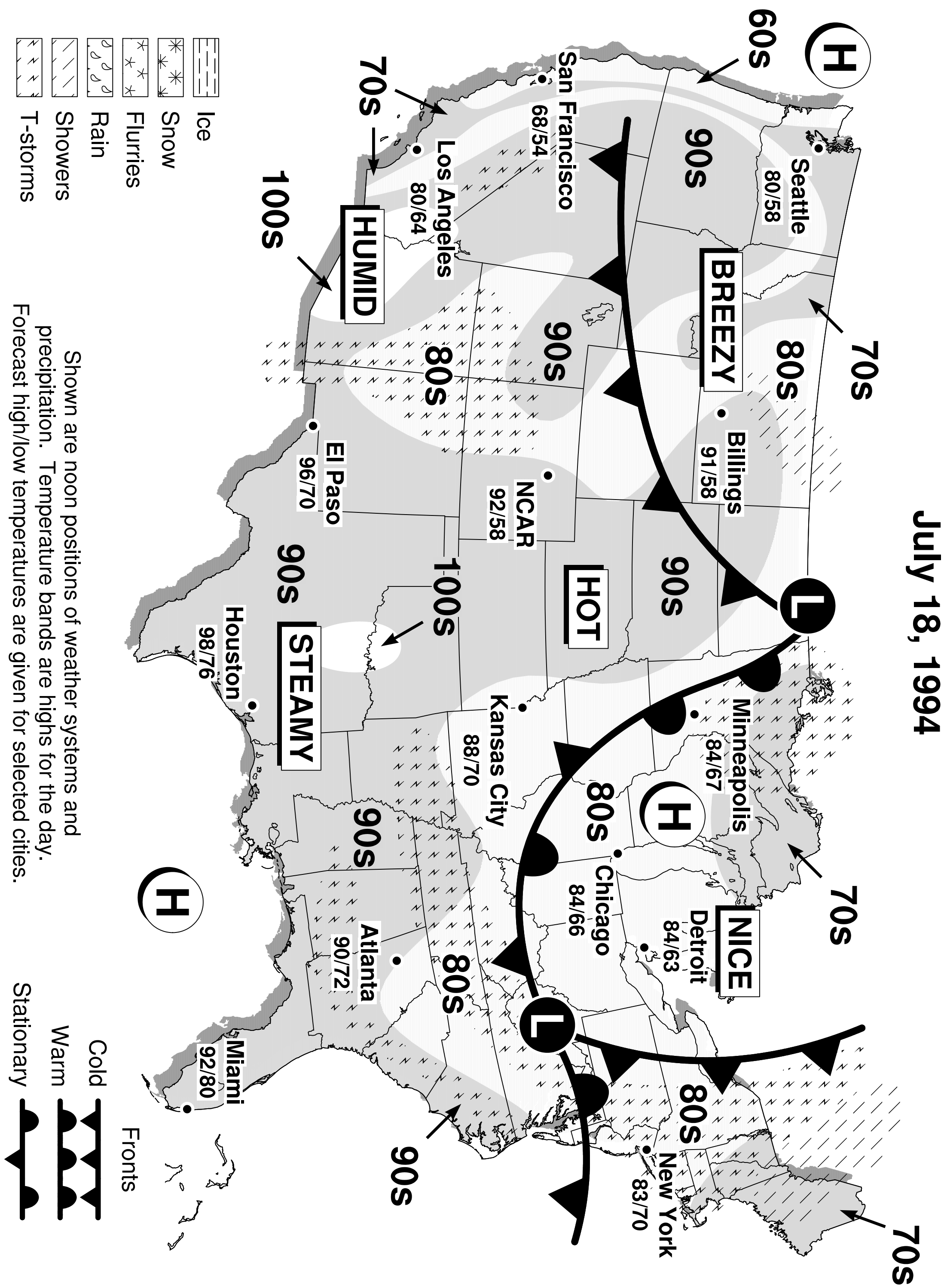
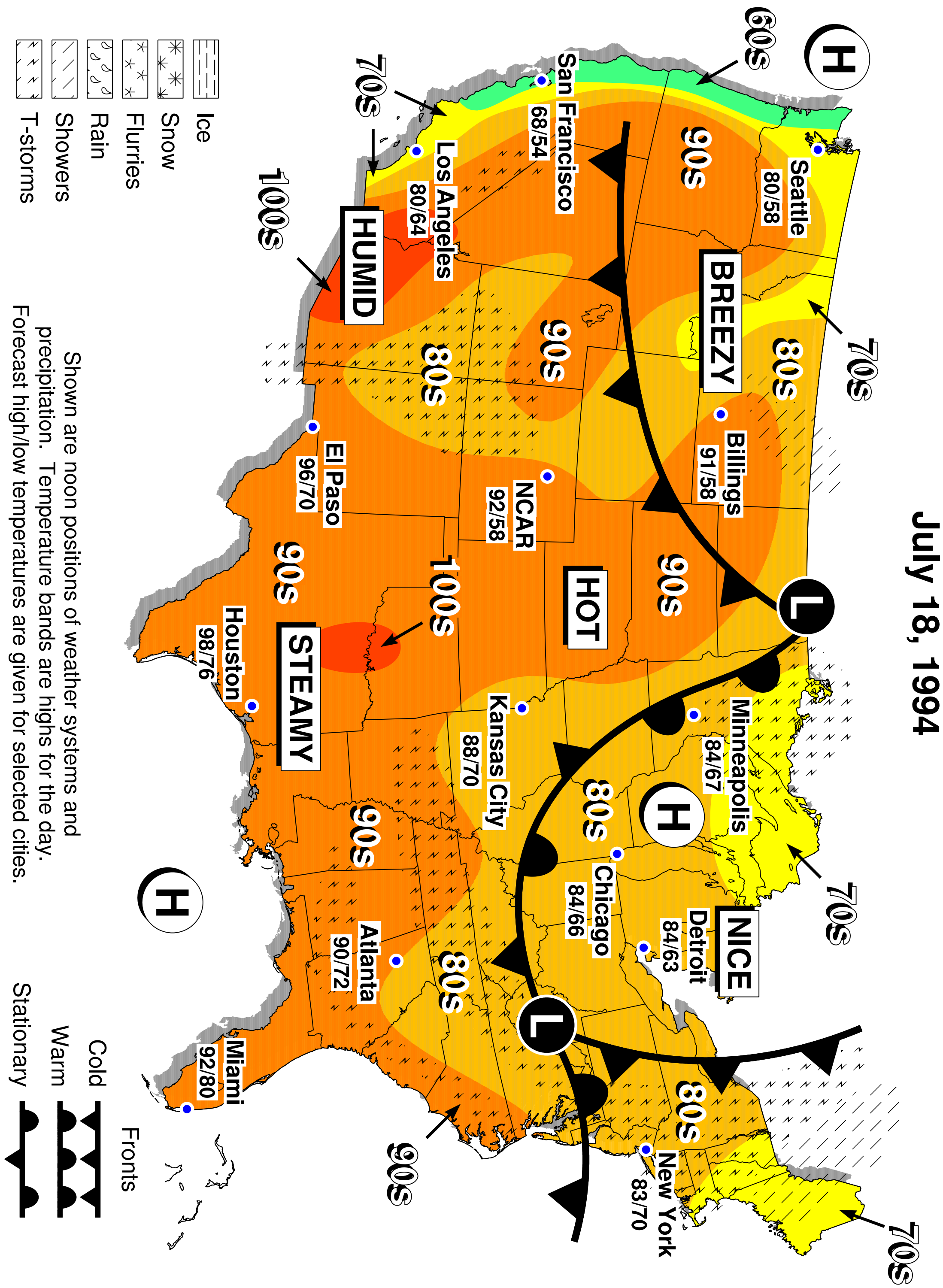


Figure 14 - ncargex example wmex14



INDEX

D

E

R

S