APPENDIX D

GEMPAK Tutorial

In Chapter 2, we saw a list of the commands available within the GEMPAK user interface. In this appendix, we will actually run, step by step, a few GEMPAK programs. It is very important that as you work through these examples, you use the on-line help files and Chapters 3 and 4 in this manual. Obviously, we will not be able to go through every GEMPAK program in this tutorial, but if you learn how to use the on-line help and the manual, you will be able to run any GEMPAK program, just by reading the available help.

To run a GEMPAK program, for example **SFLIST**, the surface listing program, simply type the name of the program. These names are case sensitive, so you must use lower case:

```
sflist
```

If your path is set up properly the program will run. If it doesn't run, try using the $GEMEXE environment variable to point to the proper directory.

```
$GEMEXE/sflist
```

Eventually, you'll probably want to set up your path to automatically search the GEMPAK bin directory for GEMPAK executable files.

Once you start SFLIST, you'll see the following information on your screen:

```
SFFILE      Surface data file
AREA        Data area
DATTIM      Date/time
SFPARM      Surface parameter list
OUTPUT      Output device/filename
IDNTYP      STNM or STID
Parameters requested: SFFILE,AREA,DATTIM,SFPARM,OUTPUT,IDNTYP.
GEMPAK-SFLIST>
```

This is the GEMPAK user interface. Any GEMPAK program will give you a similar looking screen. When you run your program, there will probably be a list of values to the right of the descriptions. This is because there is a global parameter file, which saves the most recent settings of all the variables. This is rather helpful when you are running GEMPAK in your work. Often you will find yourself using the same variable settings over and over again. With GEMPAK's "memory," you won't need to keep resetting everything.

You should now take a look at the help information for the program **SFLIST**. Either type

```
GEMPAK-SFLIST>phelp sflist
```

or look at the printed version of the SFLIST help in Chapter 4 of this manual. You will see a list of variables, a program description, and some examples. Read this section now.

After displaying help on your screen, you may want to look at the variable list again. As we learned in Chapter 2, the *display* or *list* commands will do just that.

```
GEMPAK-SFLIST>display
```

Now let's see the variables and run the program.

There are some sample data files from Hurricane Bob contained in the GEMPAK test data directory. Choose one (a surface file ending with sfc) and set the **sffile** variable to it

```
GEMPAK-SFLIST>sffile=$GEMDATA/nmcbob.sfc
```

We have used the $GEMDATA environment variable here to point to the data directory and save some typing! There are two acceptable formats for using an environment variable within a GEMPAK program. You could have also issued the following command:

```
GEMPAK-SFLIST>sffile=gemdata:nmcbob.sfc
```

You can also use the full path name of the directory.

While you're at it, take a look at the help information for **sffile** (either *phelp sffile*, or in Chapter 3 in this manual).

Next you should choose the area for which you'd like to see some data. Don't choose the entire US, since the data will all flash across the screen before you can possibly read it. There are various ways of specifying the data area. Read about them in the help information (*phelp area* or Chapter 3 in this manual). Let's choose a simple one, the geographical area around Denver.

```
GEMPAK-SFLIST>area=den
```

(If you wanted JUST Denver, you would specify **area=@den**, but you know that already from reading the help information.)

Now set the date and time variable, **dattim**. If you were looking at a current data file, you would be able to use the standard **dattim** specification on YYMMDD/HHMM (year,month,day/hour,min). Since we're using a sample data file and we don't know which dates and times are available in the file, we'll have to use a more general

specification. You can read about these (FIRST, LAST, ALL, LIST) in the **dattim** help information.

```
GEMPAK-SFLIST>dattim=last
```

You must now set the **sfparm**, or surface parameter list. This is a good time to use the manual, rather than *phelp*, since the listing of available parameters is quite long. For now, choose something simple like temperature and dewpoint.

```
GEMPAK-SFLIST>sfparm=tmpf;dwpf
```

Note the semicolon (;) delimiter.

Finally, set the **output** and **idntyp**. Don't forget to read about these variables!

```
GEMPAK-SFLIST>output=t
GEMPAK-SFLIST>idntyp=stid
```

Now, re-display all your variable settings to be sure you've specified them properly. You know from Chapter 2 that the *display* command will do that. This time, however, let's use an abbreviation. Remember, GEMPAK will allow you to abbreviate, as long as your abbreviation is unique.

```
GEMPAK-SFLIST>dis
SFFILE      Surface data file          $GEMDATA/nmcbob.sfc
AREA        Data area                  den
DATTIM      Date/time                  last
SFPARM      Surface parameter list     tmpf;dwpf
OUTPUT      Output device/filename     t
IDNTYP      STNM or STID               stid
GEMPAK-SFLIST>
```

Once you've set all the variables to what you want them to be, type *run* (or just *r*) to run the program. You will get a list of temperatures and dewpoints for the Denver area.

```
GEMPAK-SFLIST>run
STN         YYMMDD/HHMM          TMPF        DWPF
CAO         910820/0600          62.96       57.02
PGA         910820/0600          78.08       39.02
DDC         910820/0600          71.96       53.96
P28         910820/0600          75.02       51.08
1K5         910820/0600          71.06       60.98
GLD         910820/0600          69.98       60.98
DEN         910820/0600          60.08       50.00
PUC         910820/0600          62.96       34.88
GJT         910820/0600          69.98       50.00
...
Parameter requested: SFFILE,AREA,DATTIM,SFPARM,OUTPUT,IDNTYP.
GEMPAK-SFLIST>
```

Now you can either exit (type *exit*) from the program, or you can play around a bit more by listing other kinds of data for other places and times.

Eventually you will get bored by the listing program, and so next we will run a graphics program. Let's run the surface mapping program **SFMAP**. Remember your assignment, read the help information for each variable as we go along.

To start the **SFMAP** program, type

```
sfmap
```

You will notice a new thing happening. You get information on your screen about new processes being started:

```
Creating process: gplt for queue 24581
```

You will also get a list of variables:

```
AREA        Data area
GAREA       Graphics area
SATFIL      Satellite image filename(s)
RADFIL      Radar image filename(s)
SFPARM      Surface parameter list
DATTIM      Date/time
SFFILE      Surface data file
COLORS      Color list
MAP         Map color/dash/width
LATLON      Line color/dash/width/label/inc
MARKER      Marker color/type/size/width/hw
TITLE       Title color/line/title
CLEAR       Clear screen flag
PANEL       Panel loc/color/dash/width/regn
DEVICE      Device|name
PROJ        Map projection/angles/margins
FILTER      Filter data factor
TEXT        Text size/font/width/hw flag
SKPMIS      Skip missing data flag
Parameters requested: AREA,GAREA,SATFIL,RADFIL,SFPARM,
DATTIM,SFFILE,COLORS,MAP,LATLON,MARKER,TITLE,CLEAR,PANEL,
DEVICE,PROJ,FILTER,TEXT,SKPMIS.
```

Don't forget to read about the **SFMAP** program and these variables.

Let's make a station plot for Colorado. Set the variables to the following values:

```
GEMPAK-SFMAP>area=co-
GEMPAK-SFMAP>garea=co
GEMPAK-SFMAP>sfparm=skyc;tmpf;wsym;rmsl;;dwpf;p24i*100;brbk
GEMPAK-SFMAP>dattim=last
GEMPAK-SFMAP>sffile=$GEMDATA/nmcbob.sfc
GEMPAK-SFMAP>colors=1;2;3;5;6;7;11
GEMPAK-SFMAP>map=17/1/2
```

```
GEMPAK-SFMAP>latlon=
GEMPAK-SFMAP>marker=
GEMPAK-SFMAP>title=6/-2
GEMPAK-SFMAP>clear=yes
GEMPAK-SFMAP>panel=0
GEMPAK-SFMAP>device=xw
GEMPAK-SFMAP>proj=nps
GEMPAK-SFMAP>filter=yes
GEMPAK-SFMAP>skpmis=no
```

I chose the XW driver for your device. If you don't have X Windows, you should choose a different driver. Read the help information for your choices. Note the order of the *sfparm* variables determine their position around the station plot (see the help for **sfmap**).

When you're ready to plot, type run.

```
GEMPAK-SFMAP>run
```

The first thing you should see is an X Window popping up on your screen. Either bring your working window in front of the graphics window, or iconify the graphics window (don't close it!). On your working window, you should see something like this:

```
SFMAP PARAMETERS:
Data area:          co-
Graphics area name: co
Valid parameters:   SKYC  TMPF  WSYM  RMSL  BLNK  DWPF  P24I  BRBK
Parameter colors:   1     2     3     4     0     5     6     11
Time:               910820/0600
File:               $GEMDATA/nmcbob.sfc
Map:                17/1/2
Marker:
Title:              6/-2
Device:             xw
Projection:         nps
Clear screen:       YES
Filter:             YES
Filter factor:      1.00
Panel:              0
Enter <cr> to accept parameters or type EXIT:
```

This summary is your chance to see what will happen, before it actually happens. Take a close look here to be sure the settings you have are the settings you want. If you've made a mistake, that's ok, simply type *exit* and you will be returned back to the user interface. Type *display* and you can change any of the variables. On the other hand, if the information above looks good, just hit return (<*cr*>) and your map will plot.

When you're done looking at the plot, bring your working window back to the foreground, and you'll see that you're back in the user interface of **SFMAP**.

If for some reason, you think your plot doesn't look right, type *display* again to see what values your variables have. Be sure they are the same as those above.

Now, I encourage you to play around with **SFMAP**. Remember, this is Hurricane Bob data and the surface map of Colorado may be interesting, but not nearly as interesting as a map of the eastern seaboard. You may change the values of the variables as you choose and run the program over and over again, without exiting. The window can be overwritten time and time again, as long as *clear* is set to *yes*. We will talk more about *clear* later.

One other piece of advice: try not to abort GEMPAK programs with a **control-C**. GEMPAK doesn't have the kind of exception handling that will handle this gracefully.

Try the following things with **SFMAP**:

DATTIM: Try using *list*, *last*, and *first*. Try using the YYMMDD/HHMM specification. Try abbreviating it.

SFPARM: Try other parameter lists. Remember not all surface files will have all parameters available. It depends on what has been decoded. Also, remember, if you choose a large area, and a lot of surface parameters, the program is going to take a long time to draw them.

AREA/GAREA: Discover the difference between the data area (*area*) and the graphics area (*garea*). Experiment with making *area* larger than *garea* and vice versa. Experiment with the different ways to specify *area* and *garea*. Notice the difference between acceptable ways to specify *area* and *garea*. Try item #8 in Chapter 3 under *AREA (SHDR:iloval:ihival)*. Set **area=SELV:1900:5000** for Colorado. Set *area* and *garea* to the east coast entry in the geog.tbl (**area=EAST, garea=East**).

Other variables: Play with *colors*, *map*, *latlon*, *marker*, *title*, *panel*, *proj*, *filter*, *text*, and *skpmis*.

Note: You may change the map (not the color, but the map itself) from the default to any of the other available maps, by using the *$MAPFIL* variable. This variable is not displayed by the user interface, however it is available in any program that uses a map. Read the help information on *$MAPFIL* in Chapter 3. You can always reset the *$MAPFIL* back to the default as follows:

```
GEMPAK-SFMAP>$MAPFIL=hipowo.cia
```

After you've tired of playing around with the surface mapping program, it is time to exit from the program. Type *exit*:

```
GEMPAK-SFMAP>exit
```

You'll notice that your graphics X Window is still present on your workstation. With GEMPAK, you need to explicitly tell the window that you are done with it. This is

useful when you want to construct overlays. More about that later. Use the program **GPEND** to end the graphics program.

```
gpend
```

The window will pop up one last time and then disappear forever. DO NOT CLOSE THE WINDOW ANY OTHER WAY! If by mistake you do, and you do it often enough, eventually you'll run out of something called "message queues" and you'll have to clean out those message queues or reboot your workstation. See your software administrator for more details.

Let's move on to an upper air program. Let's plot a skewT diagram using **SNPROF**.

```
snprof

SNFILE      Sounding data file
DATTIM      Date/time
AREA        Data area
SNPARM      Sounding parameter list
LINE        Line color/dash/width/label
PTYPE       Plot type/h:w ratio/margins
VCOORD      Vertical coordinate type
STNDEX      Stability indices
STNCOL      Stability index color
WIND        Wind symbol/siz/wdth/typ/hdsz
WINPOS      Wind position
MARKER      Marker color/type/size/width/hw
BORDER      Background color/dash/width
TITLE       Title color/line/title
DEVICE      Device|name
YAXIS       Ystrt/ystop/yinc/lbl;gln;tck
XAXIS       Xstrt/xstop/xinc/lbl;gln;tck
FILTER      Filter data factor
CLEAR       Clear screen flag
PANEL       Panel loc/color/dash/width/regn
TEXT        Text size/font/width/hw flag
THTALN      THTA color/dash/width/mn/mx/inc
THTELN      THTE color/dash/width/mn/mx/inc
MIXRLN      MIXR color/dash/width/mn/mx/inc
Parameters requested:SNFILE,DATTIM,AREA,SNPARM,LINE,PTYPE,
VCOORD,STNDEX,STNCOL,WIND,WINPOS,MARKER,BORDER,TITLE,
DEVICE,YAXIS,XAXIS,FILTER,CLEAR,PANEL,TEXT,THTALN,THTELN,
MIXRLN.
GEMPAK-SNPROF>
```

Set the variables as follows, remembering to read about any unfamiliar ones.

```
GEMPAK-SNPROF>snfile=$GEMDATA/hrcbob.snd
GEMPAK-SNPROF>dattim=last
GEMPAK-SNPROF>area=@den
GEMPAK-SNPROF>snparm=tmpc;dwpc
GEMPAK-SNPROF>line=2;3/4;5/6
```

```
GEMPAK-SNPROF>ptype=skewt
GEMPAK-SNPROF>vcoord=pres
GEMPAK-SNPROF>stndex=show;brch
GEMPAK-SNPROF>stncol=1
GEMPAK-SNPROF>wind=bm3
GEMPAK-SNPROF>winpos=3
GEMPAK-SNPROF>marker=
GEMPAK-SNPROF>border=6
GEMPAK-SNPROF>title=6/-1
GEMPAK-SNPROF>device=xw
GEMPAK-SNPROF>yaxis=
GEMPAK-SNPROF>xaxis=
GEMPAK-SNPROF>filter=yes
GEMPAK-SNPROF>clear=yes
GEMPAK-SNPROF>panel=1/14
GEMPAK-SNPROF>text=1/2/2
GEMPAK-SNPROF>thtaln=3/4/2
GEMPAK-SNPROF>thteln=4/4/2
GEMPAK-SNPROF>mixrln=5/4/2

GEMPAK-SNPROF>dis
```

We have left some variables blank (e.g., *yaxis*). This allows GEMPAK to choose the best value for that variable. As you become more proficient with GEMPAK, you may find that sometimes this is fine, and other times you will want to override the default.

Let's run the program:

```
GEMPAK-SNPROF>r

Next plot - Time:910820/1200                    Station:  72469  DEN
Enter <cr> to accept parameters or type EXIT:
```

Hit return (<*cr*>) at this prompt to accept the parameters and plot the skewT.

```
Parameters requested: SNFILE,DATTIM,AREA,SNPARM,LINE,PTYPE,
VCOORD,STNDEX,STNCOL,WIND,WINPOS,MARKER,BORDER,TITLE,
DEVICE,YAXIS,XAXIS,FILTER,CLEAR,PANEL,TEXT,THTALN,THTELN,
MIXRLN.
GEMPAK-SNPROF>
```

You should have a nice skewT graph of Denver. Again I encourage you to experiment with **SNPROF**. In particular, you may want to try using vertical coordinates other than pressure (read the help information on *vcoord*). When you're finished with **SNPROF**, exit and don't forget to use **GPEND**.

```
GEMPAK-SNPROF>exit
gpend
```

Now we've been through an example listing program (**SFLIST**), an example mapping program (**SFMAP**) and an example plotting program (**SNPROF**). Our next and final example will use grid programs.

Let's start with the gridded data contouring program **GDCNTR**.

```
gdcntr

GLEVEL      Grid level
GVCORD      Grid vertical coordinate
GFUNC       Scalar grid
GDFILE      Grid file
CINT        Contour interval/min/max
LINE        Line color/type/width/label
MAP         Map color/dash/width
TITLE       Title color/line/title
DEVICE      Device|name
SATFIL      Satellite image filename(s)
RADFIL      Radar image filename(s)
PROJ        Map projection/angles/margins
GAREA       Graphics area
CLEAR       Clear screen flag
PANEL       Panel loc/color/dash/width/regn
TEXT        Text size/font/width/hw flag
SCALE       Scalar scale / vector scale
LATLON      Line color/dash/width/label/inc
HILO        Color/symbol/rng/rad/cnt/intp
HLSYM       HILO txt size/posn/font/wdth/hw
CLRBAR      Color/ornt/anch/x;y/ln;wd/freq
CONTUR      Subbox/smooth
SKIP        Skip_cntr/skip_plt_x;skip_plt_y
FINT        Fill interval/min/max
FLINE       Fill colors
CTYPE       Contour type:  C/F
Parameters requested: GDATTIM,GLEVEL,GVCORD,GFUNC,GDFILE,CINT,
LINE,MAP,TITLE,DEVICE,SATFIL,RADFIL,PROJ,GAREA,CLEAR,PANEL,
TEXT,SCALE,LATLON,HILO,HLSYM,CLRBAR,CONTUR,SKIP,FINT,FLINE,
CTYPE.
GEMPAK-GDCNTR>
```

Here are some new variables that we haven't seen before. Let's start by choosing the sample grid file and the grid date and time of 910819/0000F00.

```
GEMPAK-GDCNTR>gdfile=$GEMDATA/hrcbob.grd
GEMPAK-GDCNTR>gdattim=910819/0000F00
```

Note that now we are using *gdattim* rather than *dattim*.

Now let's specify the grid and the variable we want to contour.

```
GEMPAK-GDCNTR>glevel=500
GEMPAK-GDCNTR>gvcord=pres
GEMPAK-GDCNTR>gfunc=hght
```

We've chosen to contour 500 millibar heights (this grid is present in the data file; try running **gdinfo**, the grid file information program, to demonstrate this for yourself).

We leave *cint* blank to allow the program to calculate the contour interval for us. We use standard values for the other variables.

```
GEMPAK-GDCNTR>cint=
GEMPAK-GDCNTR>line=5
GEMPAK-GDCNTR>map=6/1
GEMPAK-GDCNTR>title=6/-1
GEMPAK-GDCNTR>device=xw
GEMPAK-GDCNTR>proj=nps
GEMPAK-GDCNTR>garea=usnps
GEMPAK-GDCNTR>clear=yes
GEMPAK-GDCNTR>panel=0/6
GEMPAK-GDCNTR>text=1/2/2
```

We set *scale* to *999* so that the program will automatically scale out data by an appropriate amount if necessary. We also chose the contour type to be contour lines (*c*). We could choose contour color fill as well (**ctype=c/f**). Try it on your own later.

```
GEMPAK-GDCNTR>scale=999
GEMPAK-GDCNTR>latlon=
GEMPAK-GDCNTR>skip=
GEMPAK-GDCNTR>fint=
GEMPAK-GDCNTR>fline=
GEMPAK-GDCNTR>ctype=c
```

Now let's *run* the program. Notice that the contour levels and interval are chosen by the program. You can override this if you'd like. Read about *cint* in Chapter 3 or by typing *phelp cint*.

```
GEMPAK-GDCNTR>r

Grid file: $GEMDATA/hrcbob.grd

GRID IDENTIFIER:
          TIME1     TIME2                LEVL1      LEVL2     VCORD     PARM
 910819/0000F000                           500                  PRES     HGHT

GAREA: usnps
SCALE: 0

MINIMUM AND MAXIMUM VALUES  5504.61  5959.61

LINE CONTOURS:
LEVELS:  5550.00  5600.00  5650.00  5700.00  5750.00  5800.00  5850.00
COLORS:        5        5        5        5        5        5        5
LINTYP:        1        1        1        1        1        1        1
LINWID:        1        1        1        1        1        1        1
LABEL:         1        1        1        1        1        1        1

LEVELS:  5900.00  5950.00
COLORS:        5        5
LINTYP:        1        1
LINWID:        1        1
LABEL:         1        1

Enter <cr> to accept parameters or type EXIT:
```

Hit return (<*cr*>) to see the plot.

Now, let's say that you particularly like this plot. You like all the colors and the contour intervals. In fact, you may want to look at other data using these same variable settings. You can use the *save* and *restore* commands that you learned in Chapter 2 to save these settings and restore them later. First *save* these settings.

```
Parameters requested: GDATTIM,GLEVEL,GVCORD,GFUNC,GDFILE,CINT,
LINE,MAP,TITLE,DEVICE,SATFIL,RADFIL,PROJ,GAREA,CLEAR,PANEL,
TEXT,SCALE,LATLON,HILO,HLSYM,CLRBAR,CONTUR,SKIP,FINT,FLINE,
CTYPE.
GEMPAK-GDCNTR>save gdcntr.settings
```

An ASCII text file is created with all the settings from this run of GDCNTR. Since this is a text file, you may edit it. Often this is useful if you use the save/restore file on different data sets; you want to keep the settings, but you don't want to keep the same file name or date. Edit the file and remove the entries for *gdfile* and *gdattim*.

In the example above, I chose to call this text file **gdcntr.settings**, but you can name it anything you'd like. If you don't specify a file name, the file will be given a default name of **gdcntr.nts**. You may specify a file name with a path, if you'd like to store all your save/restore files in a particular directory.

```
GEMPAK-GDCNTR>save restore/gdcntr.settings
```

Later, when you are running **GDCNTR** again and you want to retrieve these settings, simply *restore* the file.

```
Parameters requested: GDATTIM,GLEVEL,GVCORD,GFUNC,GDFILE,CINT,
LINE,MAP,TITLE,DEVICE,SATFIL,RADFIL,PROJ,GAREA,CLEAR,PANEL,
TEXT,SCALE,LATLON,HILO,HLSYM,CLRBAR,CONTUR,SKIP,FINT,FLINE,
CTYPE.
GEMPAK-GDCNTR>restore restore/gdcntr.settings
```

You may have heard about the grid diagnostic functionality of GEMPAK. It is very simple to calculate a new grid from an existing grid or grids. GEMPAK comes supplied with a wide variety of pre-programmed diagnostic functions for calculating many meteorological fields. You can read about them by typing *phelp gparm* (instead of *phelp gfunc* or *gvect*), or you can read about them in more depth by looking at Appendix B in this manual. In addition to the pre-programmed diagnostics, there are also many mathematical operators available so you can construct your own diagnostic functions.

To illustrate this functionality, let's run GDCNTR again, but this time, let's try to contour a grid that does not exist in the data file hrcbob.grd. Let's choose a simple but useful field, potential temperature. The diagnostic function for potential temperature is **thta (tmpc,pres)** (see Appendix B or gparm in Chapter 3). The arguments of this function, **(tmpc,pres)**, are the input grids required for calculating the potential temperature.

```
GEMPAK-GDCNTR>gfunc=thta(tmpc,pres)
```

Let's also set the area to a smaller region, choose a level closer to the surface, and use color contour fill this time. Numbers are used to indicate which colors are used in the plot (all colors including *line* and *map*, etc., not just those used in contour fill). There is a table listing the available colors and their corresponding numbers under the *colors* entry in Chapter 3.

```
GEMPAK-GDCNTR>garea=east
GEMPAK-GDCNTR>glevel=700
GEMPAK-GDCNTR>line=1-15/1/2/1
GEMPAK-GDCNTR>ctype=c/f
GEMPAK-GDCNTR>fline=32-18--1

GEMPAK-GDCNTR>r

Grid file: $GEMDATA/hrcbob.grd

GRID IDENTIFIER:
          TIME1          TIME2       LEVL1 LEVL2   VCORD  PARM
   910819/0000F000                    700                 PRES  THTATMPCPRES

GAREA:    east
SCALE:  0

MINIMUM AND MAXIMUM VALUES   295.67   314.74

LINE CONTOURS:
LEVELS:    296.00    298.00    300.00    302.00    304.00    306.00    308.00
COLORS:         1         2         3         4         5         6         7
LINTYP:         1         1         1         1         1         1         1
LINWID:         2         2         2         2         2         2         2
LABEL:          1         1         1         1         1         1         1

LEVELS:    310.00    312.00    314.00
COLORS:         8         9        10
LINTYP:         1         1         1
LINWID:         2         2         2
LABEL:          1         1         1

FILLED CONTOURS:
LEVELS:    296.00    298.00    300.00    302.00    304.00    306.00    308.00
COLORS: 32        31     30        29        28        27        26

LEVELS:    310.00    312.00    314.00
COLORS:  25        24     23        22

Enter <cr> to accept parameters or type EXIT:
```

Hit return (<*cr*>) and take a look at the plot. This graphic will take a bit longer to produce, since it must fill in the colors. To shorten the drawing time, you may want to set the *skip* variable to a value of 1 or 2. *Skip* will cause **GDCNTR** to skip some grid points when calculating the contours. This will shorten the compute and drawing times,

and usually still provides a nice looking plot. In cases with extremely noisy data, or the need for very precise contours, *skip* should remain at 0.

Now let's overlay some wind barbs on top of the potential temperature contours. *Exit* from **GDCNTR**, but this time, do not use **GPEND**. Instead, start the vector drawing program, **GDWIND**.

```
gdwind

GDATTIM     Grid date/time                      910819/0000F00
GLEVEL      Grid level                          700
GVCORD      Grid vertical coordinate            pres
GVECT       Vector grid
GDFILE      Grid file                           $GEMDATA/hrcbob.grd
GAREA       Graphics area                       east
SATFIL      Satellite image filename(s)
RADFIL      Radar image filename(s)
SKIP        Skip_cntr/skip_plt_x;skip_plt_y
WIND        Wind symbol/siz/wdth/typ/hdsz
REFVEC      Mag;x;y;txtsiz/font/wdth/HW;labl
MAP         Map color/dash/width                6/1
LATLON      Line color/dash/width/label/inc
PANEL       Panel loc/color/dash/width/regn     0/6
TITLE       Title color/line/title              6/-1
DEVICE      Device|name                         xw
PROJ        Map projection/angles/margins        nps
CLEAR       Clear screen flag                   yes
SCALE       Scalar scale / vector scale         999
TEXT        Text size/font/width/hw flag        1/2/2
Parameters requested:GDATTIM,GLEVEL,GVCORD,GVECT,GDFILE,
GAREA,SATFIL,RADFIL,SKIP,WIND,REFVEC,MAP,LATLON,PANEL,
TITLE,DEVICE,PROJ,CLEAR,SCALE,TEXT.
GEMPAK-GDWIND>
```

Note that many of the variables we set in the last run of **GDCNTR** are still applicable here. This is an example of the advantage of GEMPAK's "memory" between programs.

Let's set up the variables to overlay wind barbs in knots over the potential temperature contours at 700 mb. Since this is the vector drawing program (wind barbs are vectors because they have magnitude and direction) the *gfunc* variable has been replaced by *gvect*. You can read about the options for specifying *gvect* under *gparm* on-line, in Chapter 3, or in Appendix B. Specify the representation of the wind barbs (barbs, knots, color #2) with the wind variable.

```
GEMPAK-GDWIND>gvect=wind
GEMPAK-GDWIND>wind=bk2//2
GEMPAK-GDWIND>title=6/-1/700 mb Winds
```

Pay special attention to the *clear* variable. This is what determines whether a new plot is laid on top of (overlay) the old plot, or replaces the old plot. Each time we've used

this variable in the past, we've set it to *yes*, since we wanted a new plot each time. Now we set it to *no* so the wind barbs will be plotted on top of the potential temperature contours.

```
GEMPAK-GDWIND>clear=no
```

Now let's *run* the program.

```
GEMPAK-GDWIND>r

 PARAMETERS FOR GDWIND:

 Grid file: $GEMDATA/hrcbob.grd

 GRID IDENTIFIER:
          TIME1          TIME2       LEVL1 LEVL2   VCORD  PARM
   910819/0000F000                    700                  PRES  UREL

 GAREA:  east
 SCALE:    0

 MINIMUM AND MAXIMUM VECTOR MAGNITUDE     0.09    52.41

 Wind type: bk2//2

 Enter <cr> to accept parameters or type EXIT:
```

Hit return (<*cr*>) to see the overlay plot .

Now you've generated a wind barb plot overlaid on contour of potential temperature!

You may use this technique for constructing overlays from any GEMPAK graphics programs. You may overlay plots using the same program (overlay line contours of 500 mb heights on top of color fill contours of vorticity, for example), or from different programs (overlay station data on top of lifted index contours).

Take some time now to experiment with creating overlays and using the diagnostic functions listed in Appendix B.

When you are all finished running GEMPAK, be sure to use **GPEND**.

In the course of this Appendix we've learned how to run surface, upper air, and gridded programs, to list, map, plot, contour, and overlay data. This was not an exhaustive tour through GEMPAK. In fact there are many GEMPAK programs that weren't even mentioned here. However, now that you know how to run a sampling of GEMPAK programs, and you know how to use the on-line help and the manual, you should be able to find your way through any GEMPAK program.