CONPACK, A CONTOURING PACKAGE

David J. Kennison
NCAR, P.O. Box 3000, Boulder, Colorado 80307-3000
email: kennison@ncar.ucar.edu (128.117.64.4)

CONPACK, A CONTOURING PACKAGE

David J. Kennison
NCAR, P.O. Box 3000, Boulder, Colorado 80307-3000
email: kennison@ncar.ucar.edu (128.117.64.4)

1.0 PREFACE

The contouring package described in this document was written between 1986 and 1988 to replace older contouring packages available at NCAR and to make possible the use of newly available capabilities in graphics hardware and software. Since 1988, many bugs have been fixed and a number of new capabilities have been added.

2.0 INTRODUCTION

This document describes an NCAR Graphics package called CONPACK that allows a user to construct contour plots from rectangular arrays of data. CONPACK provides a sort of tool kit of FORTRAN subroutines that can be called in various combinations to draw different kinds of contour plots.

This section is intended to give an overall view of CONPACK and selected aspects of its design; it covers some details, but, in general, one should refer to the sections “SUBROUTINES” and “PARAMETERS” for detailed descriptions of subroutines and parameters mentioned. (Parameters are mentioned by name; all the names are of the form 'XXX', where XXX is a three-character mnemonic.) The section “ERROR MESSAGES” describes error messages written by CONPACK. The section “EXAMPLES” describes the examples available for CONPACK.

It is assumed that the reader is familiar with NCAR Graphics in general and has some knowledge of the packages AREAS and EZMAP/EZMAPA.

2.1 Routines Which May Be Called

The code to draw a contour plot will probably begin with several calls to set internal parameters affecting the behavior of the routines called after that. All such internal parameters have default values; only those which are to have values different from the default need to be set. Routines which may be called to set the values of parameters are as follows:

- **CPRSET** is called to reset all parameters to their default values.

- **CPSETC** is called to give a value of type CHARACTER to a parameter.
- **CPSETI** is called to give a value of type INTEGER to a parameter.
- **CPSETR** is called to give a value of type REAL to a parameter.

In general, once a parameter is given a value by a call to one of these routines, it retains that value until a similar call resets it. Thus, many of the parameter-setting calls need to be done only once and do not need to be repeated for each new contour plot.

After all required parameters have been set, the process of drawing a contour plot begins with a call to one of three initialization routines. Which one is called depends on the type of data to be contoured. The three possibilities so far are as follows:

- **CPRECT** is called if one has an acceptably “dense” rectangular array.
- **CPSPS1** is called if one has a “sparse” rectangular array and the grid is regularly spaced in X and Y; it returns a “dense” rectangular array, generated by interpolation from the original data, using bicubic splines under tension. This may be viewed as a sort of 3D smoothing. Calls to “CPSPRS” will be treated as calls to CPSPS1.
- **CPSPS2** is called if one has a “sparse” rectangular array and the grid is irregularly spaced; otherwise, it acts just like CPSPS1.

(Eventually, it is hoped, there will be another routine, called CPRAND, to handle randomly positioned data points, but that routine is not ready yet.)

Each of the routines CPRECT, CPSPS1, and CPSPS2 performs initialization required for subsequent calls to work properly. The dimensions of the data array and of real and integer workspace arrays are copied to internal variables. Pointers used to manage the real and integer workspaces are initialized. It is decided how the indices of the data array are to be mapped into X and Y coordinates used to define contour lines and how those X and Y coordinates are to be mapped to the plotter frame. If the user has not called the SPPS routine SET (or done the equivalent GKS calls), that call is done. The minimum and maximum values in the data array are found and it is decided whether or not the data field is essentially constant. If contour levels are to be chosen automatically, the parameter arrays specifying the contour levels are cleared, so that automatic selection will take place when these levels are required.

Among the internal parameters are arrays completely specifying contour levels of interest and what is to be done at each of those levels. These arrays may be used to take complete control of the contouring process; most users will probably elect not to do this, but to let CONPACK choose the levels. At this point, then, as a rule, none of these parameter arrays will have been filled. No calls need be done to fill them; if they are empty when they are needed, the required values will be chosen at that point. For certain applications, however, it is desirable to force the selection of contour levels and perhaps the character strings which are to be used as contour labels. This may be done by means of calls to one or both of the following routines:

- **CPPKCL** is called to have CONPACK pick a set of contour levels.

- **CPPKLB** is called to pick labels to be associated with those levels for which the contour lines are to be labelled and for which no labels have yet been specified.

The advantage of calling one or both of these routines is that, after each call, one can modify what the routine called has done in order to produce desired effects. For example, after CPPKCL is called, one might set elements in parameter arrays which will cause the labelled contour lines to be solid and the unlabelled contour lines to be dashed. As another example, after CPPKLB is called, one might check for the string “0” as a label and change it to “0.0”.

At this point, various other routines may be called:

- **CPBACK** is called to draw a background.
- **CPCICA** is called to add color indices to a user’s cell array. Most likely, the user will then call the GKS routine GCA to display the cell array, thereby creating a solid-filled contour plot.
- **CPCLAM** is called to add a set of contour lines to an area map. Most likely, the user will then call the AREAS routine ARSCAM to create, from the area map, a solid-filled contour plot.
- **CPCLDM** is called to draw a set of contour lines, as masked by an existing area map.
- **CPCLDR** is called to draw a set of contour lines.
- **CPCLTR** is called to trace all of the contour lines at a given level and return them to the caller.
- **CPLBAM** is called to add label boxes to an area map. Such an area map will most likely be used to prevent contour lines from passing through the labels.
- **CPLBDR** is called to draw labels for the contour plot.

Finally, to advance the frame, the user must call the SPPS routine FRAME; CONPACK won’t do it.

At any time, it is possible to retrieve the value of an internal parameter by calling one of the three following routines:

- **CPGETC** is called to get a value of type CHARACTER.
- **CPGETI** is called to get a value of type INTEGER.
- **CPGETR** is called to get a value of type REAL.

Several routines in CONPACK are not called by the user, but by CONPACK itself. The default versions of these routines, in all cases but one, do nothing; the routines exist simply to be replaced by the user. These routines are as follows:

- **CPCHCF** is called just before and just after each step in the process of writing a constant-field message (which happens when CPRECT, CPSPS1, or CPSPS2 determines that the data field being contoured is effectively constant). A user version may be supplied to change line width, color, etc.

- **CPCHCL** is called just before and just after each contour line is drawn. A user version may be supplied to change line width, color, dash pattern, etc.
- **CPCHHL** is called just before and just after each step in the process of writing the high and low labels. A user version may be supplied to change line width, color, etc.
- **CPCHIL** is called just before and just after each step in the process of writing the informational label. A user version may be supplied to change line width, color, etc.
- **CPCHLL** is called just before and just after each step in the process of writing a contour line label. A user version may be supplied to change line width, color, etc.
- **CPDRPL** may be specified as the routine to be called by CPCLDM, to draw the pieces of contour lines resulting from masking by an area map. It will draw the polyline defined by its first three arguments if and only if none of the area identifiers defined by the remaining arguments is negative. It will use a call to CURVE or a call to CURVED, depending on the current value of the parameter 'DPU'.
- **CPMPXY** is called when the parameter named 'MAP' is non-zero. Its function is to map the coordinates of points on the data grid from one coordinate system to another. The default version of this routine will handle the mapping of longitudes and latitudes into rectangular coordinates on a given EZMAP projection and the mapping of polar coordinates into rectangular coordinates. A user version may be supplied to handle other mappings.

  As of release 3.1.3 of NCAR Graphics, the default version of CPMPXY is capable of doing inverse mappings, which provides some new capabilities. Users may wish to upgrade their own versions of CPMPXY to do inverse mappings, too. See the description of the routine CPMPXY, in the section “SUBROUTINES”, for details.

Two CONPACK routines are provided to simplify the transition from CONREC to CONPACK. Each of these routines allows one to draw a complete contour plot with a single call, but they do not provide access to the full range of capabilities of the package.

- **CPEZCT** simulates the behavior of the old routine EZCNTR.
- **CPCNRC** simulates the behavior of the old routine CONREC.

By setting internal parameters prior to calling CPEZCT or CPCNRC, one may change the quality of the output, thereby simulating either the behavior of the default version, the “smooth” version, or the “super” version of CONREC. Users may wish to obtain copies of the routines CPEZCT and/or CPCNRC to use as starting points for the construction of their own CONPACK-based contouring routines.

Two additional routines are provided for use in error-recovery situations:

- **CPMVIW** may be called to transfer the current contents of an integer workspace array to a new array.
- **CPMVRW** may be called to transfer the current contents of a real workspace array to a new array.

2.2 Different Styles of Contour Plots

The design of CONPACK allows one to construct contour plots in many different styles. Which style is chosen will depend on the amount of computer time and memory available, on the capabilities of the device on which the plots are being drawn and on the intended use of the plots. Some possibilities are discussed below.

If the intent is to draw the contour plot using relatively few computer resources, the following sequence of calls will suffice:

- (1) Call parameter-setting routines.
- (2) Call CPRECT to initialize the drawing of the contour plot.
- (3) Call CPBACK to draw a simple background.
- (4) Call CPCLDR to draw the contour lines, with labels generated by a dashed-line package.
- (5) Call CPLBDR to draw an informational label below the plot and high/low labels on the plot.

Assuming that step 1 is null and that all default parameter values are used, the resulting plot will look much like what would have been drawn by CONREC. The labels won't be positioned very well and lines may be drawn through them. The characters used for the informational label and for high and low labels will be of better quality than those used for the line labels; if desired, additional resources may be saved by a parameter-setting call forcing the PLOTCHAR package to use lower-quality characters.

To produce a better-positioned set of labels, step 1 above should include a call to CPSETI setting the parameter 'LLP', which says how line labels are to be positioned, to one of the values 3 or -3; this turns off the generation of line labels by a dashed-line package called by CPCLDR and causes them to be positioned and written by the routine CPLBDR instead, using a penalty scheme which produces quite pleasing results. Contour lines will still pass through line labels positioned in this way.

Contour lines may be prevented from passing through labels drawn by CPLBDR in one of two ways, depending on the nature of the plotting device. If the device allows for the use of the GKS fill area primitive and if the result of drawing one object on top of another is that the pixels affected by drawing the second object simply change color (as happens on most terminals, but not usually on a device which exposes film, for example), then one may insert calls changing the values of the parameters 'ILB', 'HLB', and 'LLB' in such a way as to force boxes surrounding the informational label, the high and low labels, and the line labels to be filled with the background color prior to the drawing of the labels. This has the effect of preventing the contour lines from passing through the labels.

If filling the label boxes will not work, then a software technique, using routines in the utility AREAS, may be used instead. The sequence of calls will then be as follows:

- (1) Call parameter-setting routines.

- (2) Call CPRECT to initialize the drawing of the contour plot.
- (3) Call CPBACK to draw a simple background.
- (4) Call ARINAM (in AREAS) to initialize an area map.
- (5) Call CPLBAM to generate a list of label positions and to put label boxes for the labels at those positions into the area map.
- (6) Call CPCLDM to draw contour lines masked against the area map. Each line is broken into pieces lying outside label boxes and pieces lying inside label boxes and only the former are drawn.
- (7) Call CPLBDR to draw the labels in the boxes left vacant by step 6.

Some users may wish to draw a solid-filled contour plot, rather than a line plot. This is also done using routines in the utility AREAS, as follows:

- (1) Call parameter-setting routines.
- (2) Call CPRECT to initialize the drawing of the contour plot.
- (3) Call ARINAM (in AREAS) to initialize an area map.
- (4) Call CPCLAM to put contour lines into the area map.
- (5) Call ARSCAM (in AREAS) to scan the area map and to recover from it the polygons created by the lines in the area map. A user-defined routine is called to fill (or not to fill) each polygon. Filling may be done by calls to routines in the utility SOFTFILL or to the GKS routine GFA.

Labels may be written on a solid-filled contour plot; this requires adding a call to CPLBAM after step 4 and a call to CPLBDR after step 5. It may be better, though, to draw a key for the solid-filled plot. There is no routine in CONPACK to do this, but the NCAR Graphics utility LABELBAR provides such a capability.

The packages AREAS, CONPACK, and EZMAP/EZMAPA may be used cooperatively to achieve other desired effects. For example, if the contour plot being drawn represents output from an ocean model, it may be desirable to draw contours (or to fill contour bands) only over the oceans on a background drawn by EZMAP. See example 8 in the section “EXAMPLES”.

As of release 3.1.3 of NCAR Graphics, a solid-filled contour plot may be drawn in another way, using the GKS primitive “cell array”. The steps required are as follows:

- (1) Call parameter-setting routines.
- (2) Call CPRECT to initialize the drawing of the contour plot.
- (3) Call CPCICA to generate color indices in a cell array.
- (4) Call the GKS routine GCA to “paint” the cell array.

Using cell arrays has advantages and disadvantages compared to using AREAS to generate GKS “fill areas”. One disadvantage is that, if you are using your own version of the mapping routine CPMPXY, you will have to upgrade it, in the same way that the default version of the routine has been upgraded, to make it do inverse mappings. Another disadvantage is that, for the same sort of quality, a rather high-resolution cell array must be used; it may be time-consuming and space-consuming to generate and to display such a cell array; in particular, using the “zoom” capability of “idt” on a cell array doesn’t work too well. Another disadvantage of the cell-array approach is that it is more difficult (sometimes impossible) to limit the coloring of contour bands to specific areas (like, over land or over ocean), which can be done relatively easily using the fill-area approach. The major advantage of the cell-array approach is that it avoids the problems that, to some extent, accompany the use of the package AREAS. (While many of these problems have been fixed, one can still expect an occasional glitch, particularly when using certain EZMAP projections.)

2.3 Contour Level Selection

The routines CPCLAM, CPCLDM, CPCLDR, CPLBAM, and CPLBDR all generate output for specified sets of contour levels. The parameter 'NCL' specifies the number of different contour levels for which something is to be done by one or more of the above routines. The value of 'NCL' may not exceed 256. For each value of I from 1 through 'NCL', the Ith element of the parameter array 'CLV' specifies a contour level and the Ith element of each of the parameter arrays 'AIA', 'AIB', 'CLC', 'CLD', 'CLL', 'CLU', 'LLC', and 'LLT' is an associated datum, saying something about what is to be done at that contour level. Detailed descriptions of these parameter arrays are given in the section “PARAMETERS”, below; thumbnail descriptions of elements of these parameter arrays are as follows:

- Each element of the parameter array 'AIA' is an area identifier for areas “above” the contour level (in the area where data values are greater than the level). Affects CPCICA and CPCLAM.
- Each element of the parameter array 'AIB' is an area identifier for the area “below” the contour level (in the area where data values are less than the level). Affects CPCICA and CPCLAM.
- Each element of the parameter array 'CLC' is a color index for contour lines at the level. Affects CPCLDM and CPCLDR.
- Each element of the parameter array 'CLD' is a dash pattern for contour lines at the level. Affects CPCLDM and CPCLDR.
- Each element of the parameter array 'CLL' is a line width for contour lines at the level. Affects CPCLDM and CPCLDR.
- Each element of the parameter array 'CLU' is a “use flag” for the contour level, indicating whether or not contour lines are to be drawn and whether or not the lines are to be labelled. Affects CPCLDM, CPCLDR, CPLBAM, and CPLBDR.
- Each element of the parameter array 'LLC' is a color index for labels at the level. Affects CPLBDR.

- Each element of the parameter array 'LLT' is a label for the level. Affects CPLBAM and CPLBDR.

The contents of these parameter arrays may be specified completely by CONPACK, completely by the user, or a little of both. The parameter 'CLS' says whether or not CONPACK is to select contour levels and associated quantities and, if so, in what manner. The default value of 'CLS' indicates that CONPACK is to select about 16 contour levels at multiples of some nice interval, which it is to choose. By default, then, the call to CPRECT, CPSPS1, or CPSPS2 zeroes 'NCL'; during the first subsequent call which requires contour levels to have been chosen, they are chosen. The associated parameter elements are set to indicate which contour lines should be labelled, what character strings should be used as labels, which lines should be put in an area map by a call to CPCLAM, and what area identifiers should be used for areas “above” and “below” these lines.

Other values of 'CLS' may be used to change the way in which CONPACK chooses contour levels. See the detailed descriptions of 'CLS' and of the other parameters 'CIS', 'CIT', 'CIU', 'CMN', 'CMX', 'LIS', 'LIT', and 'LIU'.

The user may elect to set all of the contour levels and associated quantities. Suppose, for example, that it is desired to draw labelled solid lines for each of the values .1, .2, .3, ..., .9 and unlabelled dashed lines for each of the values .05, .15, .25, ... .95. The following code, inserted before the call to CPRECT, CPSPS1, or CPSPS2, will set the required parameters:

```
      CALL CPSETI ('CLS - CONTOUR LEVEL SELECTION',0)
      CALL CPSETI ('NCL - NUMBER OF CONTOUR LEVELS',19)
      DO 101 I=1,19
        CALL CPSETI ('PAI - PARAMETER ARRAY INDEX',I)
        CALL CPSETR ('CLV - CONTOUR LEVEL VALUE',REAL(I)/20.)
        IF (MOD(I,2).EQ.1) THEN
          CALL CPSETI ('CLU - CONTOUR LEVEL USE',1)
          CALL CPSETI ('CLD - CONTOUR LINE DASH PATTERN',21845)
        ELSE
          CALL CPSETI ('CLU - CONTOUR LEVEL USE',3)
          CALL CPSETI ('CLD - CONTOUR LINE DASH PATTERN',65535)
        END IF
  101 CONTINUE
```

In the above code, 'CLS' is zeroed to suspend the selection of contour levels by CONPACK itself. Then, 'NCL' is set to say how many contour levels are to be defined. Then, in a loop on I from 1 to 19, 'PAI' is set to tell CONPACK which element of each parameter array is to be set, the Ith element of 'CLV' is set to REAL(I)/20., which, for each I, gives one of the desired contour levels, the Ith element of 'CLU' is set to a 1 if just the line is to be drawn or to a 3 if both the line and the labels for the line are to be drawn, and the Ith element of 'CLD' is set to 21845 (octal 52525) if a dashed line is to be used or to 65535 (octal 177777) if a solid line is to be used.

Note that 'NCL' must be set prior to setting any element of 'CLV' or the associated arrays.

Note also that, when an element of 'CLV' is set, all of the associated elements of the associated arrays receive a default value. (In fact,the default element of 'CLU' is 1, and the default element of 'CLD' is a pattern specifying a solid line, so two of the calls in the code above are redundant.)

Note also that the use of CPSETI to specify the value of 'CLD' will work only if 'DPU' is positive, implying the use of DASHCHAR routines; if 'DPU' is negative, DASHPACK routines will be used and, by default, they will expect underscores, rather than apostrophes, to represent gaps, so that CPSETC calls will be needed to define the dash patterns.

In some cases, the user will want to let CONPACK choose a set of contour levels and then either add other levels of interest, modify elements of the associated parameter arrays, or both. Suppose, for example, that it is desired to have CONPACK pick the levels, that contour lines at positive levels are to be drawn in red, that contour lines at negative levels are to be drawn in blue, and that contour lines at the zero level are to be drawn in white. The following code, inserted after the call to CPRECT, CPSPS1, or CPSPS2, would do the job:

```
      ... CALLS TO DEFINE COLOR INDICES IBLU, IRED, AND IWHI ...

      CALL CPPKCL (...)
      CALL CPGETI ('NCL - NUMBER OF CONTOUR LINES',NOCL)
      DO 101 I=1,NOCL
        CALL CPSETI ('PAI - PARAMETER ARRAY INDEX',I)
        CALL CPGETR ('CLV - CONTOUR LEVEL VALUE',CLEV)
        IF (CLEV.LT.0.) THEN
          CALL CPSETI ('CLC - CONTOUR LINE COLOR INDEX',IBLU)
        ELSE IF (CLEV.GT.0.) THEN
          CALL CPSETI ('CLC - CONTOUR LINE COLOR INDEX',IRED)
        ELSE
          CALL CPSETI ('CLC - CONTOUR LINE COLOR INDEX',IWHI)
        END IF
  101 CONTINUE
```

In the code above, the routine CPPKCL is called to force CONPACK to pick a set of contour levels. Then, the value of 'NCL' that it chose is retrieved, and a loop is run from 1 to that value. On each pass through the loop, the parameter array index 'PAI' is set to tell CONPACK what element of the parameter arrays is being accessed and then one of the contour levels chosen is retrieved to the variable CLEV. Depending on the value of CLEV, the associated element of the parameter array which specifies the color index of the contour lines at that level is set to produce a blue line, a red line, or a white line.

2.4 The CONPACK Coordinate System

The mapping of a contour plot onto the plotter frame depends on two things:

- The X and Y coordinates used by CONPACK to describe the plot.
- The definitions of the current viewport and window, as specified by a call to the SPPS routine SET or by calls to GKS routines.

The first of these is discussed in this section, and the second in the next section.

By default, to describe contour lines and other objects on the contour plot, CONPACK generates X coordinates in the same range as the first subscript of the array of data being contoured and Y coordinates in the same range as the second subscript of that array. Thus, X coordinates range from 1. to REAL(M) and Y coordinates from 1. to REAL(N), where “M” and “N” are the dimensions of the data array. (The parameters 'ZDM' and 'ZDN' have the values “M” and “N”, respectively, after the call to CPRECT, CPSPS1, or CPSPS2.) Thus, the lower left corner of the plot is at (1.,1.), the upper right corner of the plot is at (REAL(M),REAL(N)), and, assuming the array to be indexed by (I,J), the center of the grid box bounded by I = 3, I = 4, J = 6, and J = 7 is (3.5,6.5).

The parameters 'XC1', 'XCM', 'YC1', and 'YCN' may be used to cause X and Y coordinates to be generated in arbitrary user ranges. For example, one could set 'XC1' = -2, 'XCM' = 2, 'YC1' = -1, and 'YCN' = 1 to generate X coordinates between -2 (at I = 1) and 2 (at I = M), and Y coordinates between -1 (at J = 1) and 1 (at J = N). Similarly, one could set 'XC1' = LON1, 'XCM' = LON2, 'YC1' = LAT1, and 'YCN' = LAT2 to generate X coordinates between longitudes LON1 and LON2 and Y coordinates between latitudes LAT1 and LAT2.

If the parameter 'MAP' is given a non-zero value, each pair of X and Y coordinates is mapped, prior to use, by a statement of the form

```
CALL CPMPXY (IMAP,XINP,YINP,XOTP,YOTP)
```

IMAP is the non-zero value of 'MAP', XINP and YINP are the unmapped (input) coordinates and XOTP and YOTP are the mapped (output) coordinates.

The default version of CPMPXY does the following mappings:

- If IMAP = 1, XINP and YINP are assumed to be the longitude and the latitude, in degrees, of a point on the globe. ('XC1', 'XCM', 'YC1', and 'YCN' will need to have been set to generate the proper values.) It is further assumed that the package EZMAP has been initialized. The EZMAP routine MAPTRA is called to obtain XOTP and YOTP, which are the coordinates of the projected point.
- If IMAP = 2, XINP and YINP are assumed to represent rho and theta in polar coordinates. (Again, 'XC1', 'XCM', 'YC1', and 'YCN' will need to have been set to generate the proper values.) XOTP and YOTP are set by the statements

```
XOTP=XINP*COS(.017453292519943*YINP)
YOTP=XINP*SIN(.017453292519943*YINP)
```

- If IMAP = any other non-zero value, XOTP = XINP and YOTP = YINP.

Thus, if the data array being contoured is such that longitude is a linear function of its first subscript and latitude a linear function of its second subscript, then to map CONPACK output onto an EZMAP background, one need only initialize EZMAP, set 'MAP' to 1, set 'XC1' and 'XCM' to the longitudes at I = 1 and I = M, set 'YC1' and 'YCN' to the latitudes at J = 1 and J = N, and arrange for the proper SET call to be done.

Similarly, if the data array being contoured is such that rho is a linear function of its first subscript and theta a linear function of its second subscript, then to map CONPACK output onto a polar coordinate background (which one would have to draw for oneself), one need only set 'MAP' to 2, set 'XC1' and 'XCM' to the values of rho at I = 1 and I = M, set

'YC1' and 'YCN' to the values of theta at J = 1 and J = M, and, again, arrange for the proper SET call to be done.

The subroutine CPMPXY may be replaced by a version which does other desired mappings. If the CONPACK routines are loaded from a binary library, this can usually be done by just compiling one's own version of the routine, so that it replaces the one from the library.

2.5 Doing the Proper Call to SET

The mapping of a contour plot onto the plotter frame is determined not only by the ranges of the "user" coordinates generated by CONPACK to locate points on the contour plot (as described in the previous section), but also by the current definitions of the "window" in the user system and the "viewport" on the plotter frame. The window and viewport may have been defined by a call to the SPPS routine SET or by calls to GKS routines; the former will be described.

A call to the SPPS routine SET has the form

```
CALL SET (XVPL,XVPR,YVPB,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)
```

All arguments are REALs except for LNLG, which is an INTEGER. The first four arguments must all be between 0 and 1, inclusive; they define a rectangular area on the plotter frame known as the "viewport". The next four arguments define a rectangular area in "user" coordinate space known as the "window". The final argument indicates whether the mapping of user coordinates into the viewport is to be linear or logarithmic in X and Y. See the documentation of the package SPPS for further details.

By default, CONPACK (specifically, one of the routines CPRECT, CPSPS1, or CPSPS2) calls SET. One may, by setting the parameter 'SET' to zero, prevent CONPACK from doing this; in that case, one must do the call for oneself or depend on some other utility (such as EZMAP) to have done it.

If CONPACK calls SET, it always uses LNLG = 1, requesting a linear-linear mapping from the window to the viewport, and it positions the viewport and window as follows: The viewport is positioned as specified by the current values of the parameters 'VPL', 'VPR', 'VPB', 'VPT', and 'VPS'. The first four of these specify the position of a "viewport area", in which the viewport is to be centered and made as large as possible; the final one says how the shape of the viewport is to be chosen. By default, the window is defined by the values XAT1, XATM, YAT1, and YATN, where XAT1 is the X coordinate corresponding to I = 1, XATM is the X coordinate corresponding to I = M, YAT1 is the Y coordinate corresponding to J = 1, and YATN is the Y coordinate corresponding to J = N. The parameters 'WDL', 'WDR', 'WDB', and 'WDT' may be used to override this default behavior and specify the exact values to be used in the SET call to define the window. This is principally useful when the parameter 'MAP' is given a non-zero value to specify that X and Y coordinates are to be mapped by calling the subroutine CPMPXY.

When the default version of CPMPXY is used and 'MAP' is set to 1, one should just set 'SET' to zero; the proper call to SET will have been done already by EZMAP.

When the default version of CPMPXY is used and 'MAP' is set to 2, one may either set 'SET' to zero and do an appropriate call to SET or leave 'SET' non-zero and set the parameters 'WDL', 'WDR', 'WDB', and 'WDT' appropriately.

Note that, as long as the parameter 'MAP' is zero, a SET call done by CONPACK will define the window in such a way as to be consistent with the ranges of the X and Y coordinates it generates. If 'MAP' is set non-zero, then 'XC1', 'XCM', 'YC1', and 'YCN' will probably have to be set to cause the unmapped X and Y coordinates to be generated in the ranges expected by CPMPXY. The call to SET must specify window limits commensurate with the mapped X and Y coordinates; if CONPACK is to do the call, then 'WDL', 'WDR', 'WDB', and 'WDT' will probably have to be set to accomplish this.

2.6 The Special-Value Parameter and Special-Value Areas

The parameter 'SPV', if non-zero, specifies a “special” value. This value may be used in the data array being contoured to indicate a missing or unreliable data point. No contours will be drawn in any box of the grid with such a special value at one or more of its four corners.

The union of all grid boxes having a special value at one or more of the four corners constitutes a set of special-value areas. The routine CPCLAM will add the edges of such areas to the area map. The routines CPCLDM and CPCLDR may be made to draw the edges of such areas (by giving a non-zero value to element “-2” of the parameter array 'CLU').

2.7 The Out-of-Range Parameter and Out-of-Range Areas

The parameter 'ORV', if non-zero, specifies an “out-of-range” value. This is only of use when the parameter 'MAP' is non-zero, specifying that coordinates are to be mapped by calling the subroutine CPMPXY. The X coordinate returned by CPMPXY may be set equal to 'ORV' to indicate that the mapped point is outside the range in which the mapping is defined.

A possible value for 'ORV', if it is to be set non-zero, is 1.E12, which has historically been returned by the EZMAP routine MAPTRN (and, now, MAPTRA) to indicate a point which is outside the area depicted by a given map projection.

The union of all points for which CPMPXY returns the out-of-range value constitutes a set of out-of-range areas. Contour lines are not traced in out-of-range areas (indeed, they cannot be). A binary-halving technique is used to extend contour lines to the very edge of such areas. The routine CPCLAM will attempt to generate and add to the area map a set of edges for such areas, and the routines CPCLDM and CPCLDR may be made to attempt to draw the edges of such areas (by giving a non-zero value to element “-3” of the parameter array 'CLU'). Currently, the edges can only be traced successfully if they are defined by continuous curves having continuous first derivatives and no points of inflection or (as of version 3.1.3) if the version of CPMPXY in use is capable of doing the inverse transformations (as is true of the default version of CPMPXY).

When contour lines are traced, if two consecutive points are out of range (in range), then the entire line segment connecting those two points is assumed to be out of range (in range). If the detail of the out-of-range areas is small enough, this assumption may cause errors. Giving the parameter 'PIC' a non-zero value will cause more points to be examined along each such line segment, thus curing the problem. For similar reasons, the algorithms used to trace the edge of the grid, the edges of special-value areas, and the edges of out-of-range areas may fail. Giving the parameter 'PIE' a non-zero value will cause these algorithms to use a finer grid, thus, again, curing the problem.

2.8 2D Smoothing of Contour Lines

Each of the routines which generates contour lines (CPCLAM, CPCLDM, and CPCLDR and the internal routine which positions labels along the lines using either the penalty scheme or the regular scheme) may be caused to smooth those lines, using cubic splines under tension. Giving the parameter 'T2D' a non-zero value causes this smoothing to be done. The absolute value of 'T2D' specifies the tension to be used; values near zero (.001, for example) yield approximately cubic splines (which may give very “loopy” curves), and large values (15., for example) yield nearly polygonal curves. Using very large values (50., for example) can cause overflow in the routines that do the smoothing.

Note that, since each contour line is smoothed separately, there is no way to absolutely ensure that it will not cause adjacent contour lines to cross each other; one must experiment with the tension to reduce the probability of that to a minimum. A reasonable value to start with is 2.5.

If 'T2D' is negative, smoothing is done prior to the coordinate mapping, if any, implied by the setting of the parameter 'MAP'. If 'T2D' is positive, smoothing is done after the mapping.

The parameter 'PIC' says how many points are to be interpolated between each pair of points defining the contour line, before smoothing. If 'PIC' is given a non-zero value when the 2D smoother is turned on, the effect is to constrain the smoothed curves to more closely approximate the original polygonal lines.

The parameter 'SSL' specifies the distance between points used to draw the smoothed contour lines. It is expressed as a fraction of the width of the window in the coordinate system in which the smoothing is being done.

2.9 3D Smoothing of Contour Lines

Either of the routines CPSPS1 or CPSPS2 fits a smooth surface to a sparse array of data (using bicubic splines under tension) and then samples that surface to get a dense array of data, which is returned to the user and may be used in subsequent calls to produce the contour plot. This is quite expensive in terms of computer time and space used, but it does produce very smooth contour lines which are guaranteed not to cross each other.

The data in the dense array returned by CPSPS1 or CPSPS2 may have a larger range than the original data in the sparse array (the lows may be lower and the highs higher). If the data represent a physical quantity whose value must fall inside a fixed range, the use of CPSPS1 or CPSPS2 may be inappropriate. In this case, the user should either do his/her own interpolation to a dense grid or take action to modify the contents of the dense array before continuing to draw the contour plot. If the latter is done, CPRECT must be the first routine called after the array is modified.

2.10 Dash Package Use by CONPACK

By default, CONPACK uses routines from the old dash package called DASHCHAR. By using the appropriate flags on the “ncargf77” command line (or. if “ncargf77” is not being used, by specifying a different set of libraries on a link/load command), one can substitute one of the other members of the family (DASHSMTH or DASHSUPR).

As of version 4 of NCAR Graphics, one can tell CONPACK to use routines from a new dash package, called DASHPACK, which is intended to replace all of the older packages. One does this by giving the internal parameter 'DPU' a negative value in place of the default positive value. The new dash package offers many advantages over the older ones; for a full description of it, see the programmer document called “DASHPACK, A SOFTWARE PACKAGE FOR DRAWING DASHED LINES”.

When DASHPACK is used, the following considerations apply:

- Before changing any internal parameter of DASHPACK, the CONPACK routines CPCLDM and CPCLDR save its current value, so that it can be restored before control is returned to the user.
- Because CONPACK does its own 2D smoothing, it always suspends 2D smoothing by DASHPACK routines. Internal parameters of CONPACK determine the dash pattern, the character size, and the gap/solid width to be used by DASHPACK. Therefore, user changes in the DASHPACK parameters 'DPS', 'DPT', 'TCS', 'WOC', 'WOG', and 'WOS' are overridden by CONPACK and have no effect (except when they are done within the user callback routine CPCHCL). All other DASHPACK parameters may be changed by the user and will have the effect described in the DASHPACK programmer document.
- By default, dash patterns must be specified with dollar signs representing solids (which is the same as for the older packages), but with underscores (instead of apostrophes) representing gaps. (Alternatively, one can tell DASHPACK to expect apostrophes to represent gaps, but the underscores are more intuitive and easier to use in FORTRAN.)
- If the DASHPACK parameters 'CHB', 'CHG', and/or 'CHS' are changed by the user, CONPACK dash patterns must be changed to match.
- By default, DASHPACK calls PLCHHQ to write numeric labels along the dashed line. If it is desired to use PLCHMQ or PLCHLQ, one sets the DASHPACK parameter 'PCF' appropriately.

- If it is desired that numeric labels be made to bend with the contour lines that they label (which is possible with the new package), this can be done by setting the DASHPACK flag 'SCF' non-zero only if the numeric labels contain no function codes for PLCHHQ; if the numeric labels contain such function codes, the numeric labels must be modified by the user's code to include "break" characters. The example "cpex12" shows how to do this.

2.11 Hachuring of Contour Lines

The parameters 'HCL', 'HCF', and 'HCS' may be used to request that contour lines be "hachured" in one of various ways. "Hachuring" is the process of drawing "hachures": short line segments perpendicular to the contour and usually drawn on the "downslope" side of it.

Note: When hachuring is activated, it may be necessary to increase the value of the internal parameter named 'RWC'. See the descriptions of the parameters 'HCF' and 'RWC' for more information.

2.12 Labels

Two or three different types of labels are written by a call to the routine CPLBDR: an informational label, high and low labels, and contour-line labels (the latter only if ABS('LLP') is greater than 1). Boxes surrounding these labels may be added to an area map by calling the routine CPLBAM; the purpose of this will probably be to prevent contour lines drawn by a subsequent call to the routine CPCLDM from passing through the labels or to prevent the label boxes from being filled or colored by a subsequent call to the routine ARSCAM (in the package AREAS).

When a constant field was detected by the initial call to CPRECT, CPSPS1, or CPSPS2, a fourth type of label may be written by CONPACK routines. In this case, a call to CPLBDR will write a constant-field label, warning of the situation, in place of the labels it would normally write. A call to CPLBAM will add the label box for the constant-field label to the area map, instead of the label boxes for the other labels. Calls to CPCLDR and CPCLDM which would normally draw contour lines will write the constant-field label instead.

The appearance of all of these labels may be determined in detail by setting parameters:

- The appearance of the informational label is affected by the values of the parameters 'ILA', 'ILB', 'ILC', 'ILL', 'ILP', 'ILS', 'ILT', 'ILW', 'ILX', and 'ILY'.
- The appearance of the constant-field label is affected by the values of the parameters 'CFA', 'CFB', 'CFC', 'CFL', 'CFP', 'CFS', 'CFT', 'CFW', 'CFX', and 'CFY'.
- The appearance of high and low labels, which are placed at the positions of the highs and lows they label, is affected by the values of the parameters 'HIC', 'HIT', 'HLA', 'HLB', 'HLC', 'HLL', 'HLO', 'HLS', 'HLT', 'HLW', 'HLX', 'HLY', 'LOC', and 'LOT'.

- The appearance of contour-line labels, which are placed along the contour lines they label, as determined by the parameter 'LLP', is affected by the values of the parameters 'LLA', 'LLB', 'LLC', 'LLL', 'LLO', 'LLS', 'LLT', and 'LLW'. (This is true only when the value of 'LLP' is not equal to 1. When 'LLP' is equal to 1, contour-line labels are generated as part of the dash pattern for the contour lines and are affected only by the values of the parameters 'CLC', 'CLL', 'CLU', 'DPS', and 'DPU'.)

In all of the above parameter names, a suffixed 'A' means "angle", 'B' means "box flag", 'C' means "color index", 'L' means "line width", 'S' means "size of characters", 'T' means "text of label", 'W' means "white space width", 'X' means "X coordinate", and 'Y' means "Y coordinate". The 'O' in 'HLO' means "overlap", while the 'O' in 'LLO' means "orientation".

All labels are written by means of calls to the character-plotting routine PLCHHQ, in the package PLOTCHAR. The angle, in degrees, at which a label is written is determined by the value of the parameter 'xxA' (and, if it is a contour-line label, by the value of the parameter 'LLO'). The box flag 'xxB' determines whether or not, prior to writing the label, a box surrounding it is filled, and whether or not, after writing the label, the edge of the box is drawn. If the box is filled, it is done using the color index specified by the parameter 'LBC'; if the edge of the box is drawn, it is done using the color index, if any, chosen for the label itself, which is determined by the value of the parameter 'xxC'. The line width to be used in drawing the box is determined by the value of the parameter 'xxL'. The size (width) of the characters is determined by the value of the parameter 'xxS'. The text of the label is determined by the value of the parameter 'xxT'; usually, this string may contain embedded substrings of the form 'xxx', which are to be replaced by the value of the quantity specified by the three-character mnemonic 'xxx'. The width of the "white space" to be left around the label (which defines the dimensions of the box around it) is determined by the value of the parameter 'xxW'.

2.13 Positioning of Labels on Contour Lines

CONPACK provides three different ways to position labels on contour lines. The parameter 'LLP' is set by the user to choose one of the three.

- When ABS('LLP') = 1, contour lines are labelled by CPCLDR or CPCLDM; they do it by calling a dashed-line package with a dash pattern which includes the desired label, so that it will appear at intervals along the line. CPLBAM may be called, but it will not add label boxes for contour line labels to the area map; because of this, there is no way to keep other contour lines from passing through the labels. This is the fastest choice, but the labels may not be well positioned (they may overlap each other, for example) and there may be no easy way to force all the labels to be written at the same angle. (Also, as mentioned in the previous section, labels that are generated in this manner are not affected by changes in the various internal parameters with names of the form 'LLx'.)
- When ABS('LLP') = 2, a "regular" scheme is used to position the contour line labels. The nth label on each labelled contour line will be at a distance RC1 + RC2 * (n-1) + RC3 * Rn units (in the fractional coordinate system) from the beginning of the line,

where 'RC1', 'RC2', and 'RC3' are internal parameters and “Rn” is a random number between -1 and 1. Labels which would overlap the edge of the viewport or each other are omitted. The contour line labels are drawn by a call to either of the label-drawing routines CPLBDR or CPLBDM; in the latter case, CPLBAM may be called to add label boxes for contour-line labels to the area map, so contour lines can be prevented from passing through them. This scheme is better than the first one, but the contour-line labels are still not very well positioned. It is somewhat more expensive in terms of computer time and space used.

- When ABS('LLP') = 3, a “penalty” scheme, based on one developed by Starley Thompson and Phil Rasch, is used. The contour lines are traced in order of increasing contour level. At each point along each line, several tests are performed; if the point fails any of the tests, no label is put there. At the remaining points on the line, a penalty function is evaluated. A label is placed at the point where the penalty function has the smallest value. This process is repeated for each line until it results in the placement of no further labels. The contour line labels are drawn by a call to either of the label-drawing routines CPLBDR or CPLBDM; in the latter case, CPLBAM may be called to add label boxes for contour-line labels to the area map, so contour lines can be prevented from passing through them. This scheme gives by far the best results, but it is also the most expensive in terms of computer time and space used.

When 'LLP' is 2 or 3, smoothing, if any, implied by a non-zero value of 'T2D' is suspended during placement of contour line labels. This saves a good deal of computer time and space and provides almost as good a set of labels. To leave smoothing turned on during placement of contour line labels, use 'LLP' = -2 or -3.

2.14 Details of the Penalty Scheme for Positioning Contour Line Labels

The penalty scheme is controlled by the ten parameters 'PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PW1', 'PW2', 'PW3', and 'PW4'. (The rationale behind the names is that the first six parameters are characterized as “constants” and the remaining four as “weights” for the four terms in the penalty function.)

A point P on a contour line will be rejected as the center point of a label under any of the following conditions:

- (1) If there are fewer than three points on the line.
- (2) If the point P is too close to the center point of any other label on the current line, where the meaning of “too close” is defined by 'PC6'.
- (3) If a label at the point P would extend outside the current viewport.
- (4) If a label at the point P would overlap any previous label.
- (5) If the estimated gradient of the field being contoured is too large at the point P, where “too large” is defined by the value of 'PC1', and 'PW1', the weight of the first term in the penalty function, is non-zero.

- (6) If the estimated number of contour lines crossing a label at the point P is too large, where “too large” is defined by the value of 'PC2', and 'PW2', the weight of the second term in the penalty function, is non-zero.
- (7) If the cumulative change in direction of the contour line within a circle covering a label at the point P is too large, where “too large” is defined by the value of 'PC3', and 'PW3', the weight of the third term in the penalty function, is non-zero.

The penalty function computed at each remaining point has the form

```
 PFUN = PW1 * GRAD / (GRAV+PC1*GRSD)            GRADIENT TERM
+       PW2 * ENCB / PC2                        NUMBER-OF-CONTOURS TERM
+       PW3 * CDIR / PC3                        CHANGE-IN-DIRECTION TERM
+       PW4 * MIN (1-EXP(-((D(I)-PC4)/PC5)**2)) OPTIMUM-DISTANCE TERM
```

The first term of the penalty function becomes larger in high-gradient regions. GRAD is the estimated gradient at the point P, GRAV is the average gradient over the whole field being contoured, and GRSD is the standard deviation of the estimated gradients over the whole field. The parameter 'PC1' specifies how far from the norm gradients are allowed to wander, as a multiple of the standard deviation. Condition 5 above implies that, for points at which the penalty function is computed, either 'PW1' is zero or GRAD is less than or equal to GRAV+PC1*GRSD.

The second term of the penalty function becomes larger as ENCB, the estimated number of contour bands crossing a label at the point P, increases. The parameter 'PC2' specifies the largest number of crossing bands allowed. Condition 6 above implies that, for points at which the penalty function is computed, either 'PW2' is zero or ENCB is less than or equal to 'PC2'.

The third term of the penalty function becomes larger as CDIR, the cumulative change in direction of the contour line in a circular region centered at the point P and with a radius equal to half the larger dimension of the label, increases. The parameter 'PC3' specifies the largest such cumulative change allowed, in degrees. Condition 7 above implies that, for points at which the penalty function is computed, either 'PW3' is zero or CDIR is less than or equal to 'PC3'.

The fourth term of the penalty function becomes larger as the distance of the point P from the centers of all labels previously placed on other contour lines deviates from an optimum value specified by the user. D(I) represents the distance to the Ith such label center. The minimum is taken over all values of I. The parameter 'PC4' is the user-specified optimum distance, specified as a fraction of the width of the current viewport. If the point P is exactly 'PC4' units away from some previous label, then “MIN(1-EXP(...))” will have the value 0; otherwise, it will be non-zero. The parameter 'PC5' is the “folding distance”, specified as a fraction of the width of the current viewport; as its value decreases, the function “1-EXP(...)” develops a sharper spike at D(I) = 'PC4'.

Thumbnail descriptions and default values of all the user-settable parameters are given below:

```
'PC1'=1.   MULTIPLIER OF THE STANDARD DEVIATION OF THE GRADIENTS
'PC2'=5.   MAXIMUM NUMBER OF CROSSING CONTOUR BANDS
```

```
'PC3'=60.  MAXIMUM CUMULATIVE CHANGE IN DIRECTION OF THE CONTOUR LINE
'PC4'=.05  OPTIMUM DISTANCE, AS A FRACTION OF THE WIDTH OF THE VIEWPORT
'PC5'=.15  FOLDING DISTANCE, AS A FRACTION OF THE WIDTH OF THE VIEWPORT
'PC6'=.30  MINIMUM DISTANCE BETWEEN LABELS ON THE SAME CONTOUR LINE, AS
           A FRACTION OF THE WIDTH OF THE VIEWPORT
'PW1'=2.   WEIGHT OF THE GRADIENT TERM
'PW2'=0.   WEIGHT OF THE NUMBER-OF-CONTOURS TERM
'PW3'=1.   WEIGHT OF THE CHANGE-IN-DIRECTION TERM
'PW4'=1.   WEIGHT OF THE OPTIMUM-DISTANCE TERM
```

2.15 Numeric Labelling Controls

Numeric labels are used in various ways in the annotation of a CONPACK plot. Contour line labels created by CONPACK are purely numeric; each gives the value of the contour level for the line. The text of the informational label is specified by the value of the character-string parameter 'ILT'; it may contain sub-strings like 'CIU', 'CMN', 'CMX', 'SFU', 'ZMN', and 'ZMX', for each of which is to be substituted a string representing the numeric value of some quantity of interest. Similarly, the labels for highs and lows are specified by the character-string parameters 'HIT', 'LOT', and/or 'HLT', which may contain the substring 'ZDV', for which is to be substituted a string representing the numeric value of the high or low.

Seven CONPACK parameters have to do with the generation of pleasing numeric labels: 'NEL', 'NET', 'NEU', 'NLS', 'NLZ', 'NOF', and 'NSD'. These are described in detail in the section “PARAMETERS”.

The interaction between the parameters 'NLS' and 'NSD' can be confusing. I was asked the following question:

> “I used contour levels from 0 to 100 and set the value of CONPACK’s 'NSD' parameter to “-2”. The contours were labelled with only ‘000’s. If I changed the value of 'NSD' to ‘-3’, I got ‘nice’ contour labels like ‘20’, ‘40’, etc. My question is this: Is the behavior undefined if you end up with more significant digits than you specified with the NSD parameter?”

I answered as follows:

> “Well, no, it’s not undefined; it just isn’t described very well anywhere. Run the following program and refer to the plot it produces while reading the subsequent discussion:

```
      PROGRAM TESTIT
        DIMENSION ZDAT(101,101),RWRK(10000),IWRK(10000)
        CALL OPNGKS
        CALL GSCLIP (0)
        CALL CPSETI (‘DPU - DASH PATTERN USE FLAG’,1)
        CALL CPSETI (‘CLS - CONTOUR LEVEL SELECTION FLAG’,100)
        CALL CPSETI (‘CIS - CONTOUR INTERVAL SPECIFIER’,1)
        CALL CPSETI (‘LIS - LABEL INTERVAL SPECIFIER’,1)
        CALL CPSETI (‘CMN - CONTOUR LEVEL MINIMUM’,1)
        CALL CPSETI (‘CMX - CONTOUR LEVEL MAXIMUM’,100)
C------>CALL CPSETI (‘NLS - NUMERIC LEFTMOST SIGNIFICANT DIGIT’,0)
        CALL CPSETI (‘NSD - NUMBER OF SIGNIFICANT DIGITS’,-2)
C------>CALL CPSETI (‘NSD - NUMBER OF SIGNIFICANT DIGITS’,-6)
        DO 102 I=1,101
```

```
          DO 101 J=1,101
            ZDAT(I,J)=(REAL(I)+REAL(J))/2.-.5
101       CONTINUE
102     CONTINUE
        CALL CPRECT (ZDAT,101,101,101,RWRK,10000,IWRK,10000)
        CALL CPCLDR (ZDAT,RWRK,IWRK)
        CALL CPLBDR (ZDAT,RWRK,IWRK)
        CALL FRAME
        CALL CLSGKS
        STOP
      END
```

"The parameter 'NLS' determines what is to be considered the leftmost significant digit in a number; its default value is such that 'the digit in the same digit position as the leftmost non-zero digit of the largest number (in absolute value) in the data field being contoured is to be considered the leftmost significant digit'. The largest number in my data field is 100.5; its leftmost significant digit is in the hundreds position. Thus, if the number to be converted for output is '9', its leftmost significant digit is considered to be the '0' in '009'. Then, since 'NSD' is set to '-2', saying that I want precisely two significant digits shown, with all the others being set to zero, I get '000'; I output the whole thing to warn the user that his number is not well represented. If the number to be converted is '45', I get '040', but, in this case, I trim off the leading zero, leaving just '40'. If the number to be converted is '100', I get '100'."

"If I uncomment the first commented-out statement, so as to use 'NLS' = 0, so that 'the leftmost non-zero digit of a number is to be considered its first significant digit', the 000's go away, but now I get '9.0' where I probably wanted '9', which some find objectionable."

"If instead I uncomment the second commented-out statement, so as to leave 'NLS' = 1, but use 'NSD' = -6, I get just what I want: 1 digit for numbers less than 10, 2 digits for numbers from 10 to 99, and 3 digits for the number '100'."

"The bottom line, really, is that one shouldn't use a value of 'NSD' which has too small an absolute value; if you do, the 000's start showing up."

Hopefully, playing with the above program will help the reader to understand the use of the parameters 'NLS' and 'NSD'.

2.16 Choosing a Scale Factor

It is possible to specify a scale factor by which field values are to be divided before conversion to a character form for display as a numeric label.

The parameter 'SFS' says how the scale factor is to be chosen. If it is given a value greater than zero, that value is the desired scale factor. If 'SFS' is given a value less than or equal to zero, CONPACK is directed to choose a scale factor to use, in one of five different ways. The value of the parameter 'SFU' may be retrieved by the user; it specifies the scale factor which has been selected for use.

The value of the scale factor may be displayed as a part of the informational label. This is done by embedding the substring 'SFU' in the string which gives the value of the parameter 'ILT'.

The default value of 'SFS' is 1, which essentially specifies that no scale factor is to be used.

2.17 Constant Field Detection

Each of the routines CPRECT, CPSPS1, and CPSPS2 checks for a field which is essentially constant. When such a field is found, the parameter 'CFF' is set non-zero; otherwise, 'CFF' is zeroed. The value of 'CFF' may be retrieved by the user.

When 'CFF' is non-zero, a call to one of the routines CPCLDM or CPCLDR will not cause any contour lines to be drawn; instead, the constant-field label will be written. The edge of the grid, the edges of special-value areas, and the edges of out-of-range areas may still be drawn.

Similarly, when 'CFF' is non-zero, a call to the routine CPLBDR will write the constant-field label instead of the labels which would normally be written, and a call to the routine CPLBAM will put the label box for the constant-field label into the area map instead of the label boxes for the normal set of labels.

2.18 Workspace Management

Many of the routines in CONPACK require one or more workspace arrays, some of type REAL and some of type INTEGER. Some routines require more workspace under some conditions than under other conditions. (For example, when the contour lines at a given level are traced, an integer workspace is used to keep track of where those contour lines were, so as to avoid tracing any of them more than once. The amount of space required for this is larger for a complicated contour plot than it is for a simple one.) Building internal workspaces into a package causes problems; to enlarge such a workspace, the source code of the package must be modified, which is unacceptable. Providing each routine with arguments specifying all of the separate workspaces that it requires leads to complicated and error-prone code.

The workspace management scheme used in CONPACK is as follows: The user defines one workspace array of type REAL and another of type INTEGER. In the call to CPRECT, CPSPS1, or CPSPS2, which initializes the drawing of a contour plot, these arrays appear as arguments (called RWRK and IWRK), together with arguments specifying their lengths (LRWK and LIWK). In subsequent calls to other CONPACK routines which require workspaces, the same arrays appear as arguments, but the lengths do not. The CONPACK routines cooperate in using these arrays in such a way as not to interfere with one another. Dynamic enlargement of one workspace at the expense of another becomes possible and the probability of running out of space is reduced.

In general, it is safest not to use the workspace arrays for other purposes between one call to a CONPACK routine and the next (unless the next is to one of the routines CPRECT, CPSPS1, or CPSPS2, which initialize the workspace pointers). At the moment, there is only one case in which the contents of the arrays are assumed to be preserved intact: If labels are being positioned using either the “regular” scheme or the “penalty” scheme, the list of label positions is created in the workspace arrays when it is first required and is assumed untouched thereafter. Other cases may arise as a result of further development of the package, however.

It is possible to find out how much space has been used in each of the workspace arrays. The parameters 'IWU' and 'RWU' are zeroed by a call to CPRECT, CPSPS1, or CPSPS2 and are updated thereafter to reflect maximum usage of space in the arrays. Thus, one might give the arrays large dimensions, create a typical contour plot, retrieve the values of 'IWU' and 'RWU' to see how much space was actually used, and then reduce the dimensions to more reasonable values.

Workspace usage by some routines cannot be predicted exactly. Upper bounds can be computed, but they may be rather large. For this reason, it may not be possible to absolutely ensure that enough workspace will be available for a given call. Therefore, there is a parameter, called 'WSO', which says what to do when a workspace overflow occurs. The four possibilities are as follows: to terminate after printing an error message, to continue running after printing an error message (this is the default), to just continue running without printing anything, or to do a recoverable-error call to SETER and then continue running without printing anything. Of course, in the latter two cases, incomplete plots may result. It is possible to find out whether or not a workspace overflow has occurred during a given call to a CONPACK routine; this is done by retrieving the values of the parameters 'IWU' and 'RWU' and comparing them with the dimensions of the workspace arrays IWRK and RWRK.

The following information may be of value in attempting to estimate how much real and integer workspace will be required by each of the user-callable routines of CONPACK. First, define the following quantities:

- NWRC = 2*'RWC' if 2D smoothing of contour lines is turned off and 7*'RWC' if 2D smoothing of contour lines is turned on.
- NWCL = 100*((NHCL+99)/100), where NHCL is the number of places where a contour line crosses a horizontal segment of the grid; if contour lines are not traced, then NWCL = 0.
- NWES = 100*((NHES+99)/100), where NHES is the number of horizontal segments in the edges of the special-value areas; if these edges are not traced, then NWES = 0.
- NWEO = 100*((NHEO+99)/100)), where NHEO is the number of horizontal segments in the grid used to trace the edges of the out-of-range areas which cross such an edge; if these edges are not traced, then NWEO = 0.
- NWL1 = 100*((4*NLBS+99)/100), where NLBS is the total number of labels written (the informational label + high/low labels + line labels).

- NWL2 = 100*((4*NIHL+99)/100), where NIHL is the number of informational labels written (0 or 1), plus the number of high/low labels written.
- NWGA is zero if the penalty scheme for positioning contour labels is not being used (that is, if ABS('LLP') is not equal to 3) or if both the weights 'PW1' and 'PW2' are zero; otherwise, it has the value 'RWG' (1000, by default). In this space is constructed an array of gradients which are used in the execution of the penalty scheme.

Then:

- CPCLAM uses real workspace of length NWRC and integer workspace of length MAX (NWCL,NWES,NWEO).
- CPCLDM uses real workspace of length 2*'RWM' + NWRC and integer workspace of length 2*'IWM' + MAX (NWCL,NWES,NWEO).
- CPCLDR uses real workspace of length NWRC and integer workspace of length MAX (NWCL,NWES,NWEO).
- CPLBAM uses real workspace of minimum length 10. If contour-line labels are positioned using either the regular scheme or the penalty scheme and the list of positions is not defined when CPLBAM is called, then more real workspace is required, of length NWRC + NWL1 + NWL2 + NWGA.
- CPLBDR uses no real workspace unless contour-line labels are positioned using either the regular scheme or the penalty scheme and the list of positions is not defined when CPLBDR is called, in which case more real workspace is required, of length NWRC + NWL1 + NWL2 + NWGA.
- CPSPS1 uses real workspace of length 100*((NSPV+99)/100)+3*MSPS*NSPS+ MAX(MSPS+NSPS+NSPS,4*IZDM), where NSPV is the number of special values in the sparse array, MSPS and NSPS are the dimensions of the sparse array, and IZDM is the first dimension of the dense array. It also uses integer workspace of length 100*((NSPV+99)/100), where NSPV is the number of special values in the sparse array.
- CPSPS2 uses real workspace of length 100*((NSPV+99)/100)+3*MSPS*NSPS+ MSPS+NSPS+NSPS, where NSPV is the number of special values in the sparse array, and MSPS and NSPS are the dimensions of the sparse array. It also uses integer workspace of length 100*((NSPV+99)/100), where NSPV is the number of special values in the sparse array.

Routines not mentioned above use no workspace, with the following exception: In the routine CPCNRC, the intent of which is to enable the user to produce a complete contour plot with a single call (as was done by the routine CONREC), are defined a real workspace array of length 5000, an integer workspace array of length 2000, and an area map array of length 12000.

2.19 The Character-Width Multiplier

The parameter 'CWM' is used as a multiplier for all character widths and similar quantities. This makes it possible to scale all such quantities up and down simultaneously. The default value of 'CWM' is 1.

2.20 Searching for Highs and Lows

The gridpoint (I,J) is defined to be the position of a high if and only if

- I is neither 1 nor 'ZDM'.
- J is neither 1 nor 'ZDN'.
- ZDAT(I,J) is not a special value.
- ZDAT(I,J) is greater than ZDAT(K,L) for all values of K and L such that K is greater than or equal to MAX(1,I-NX) and less than or equal to MIN('ZDM',I+NX), L is greater than or equal to MAX(1,J-NY) and less than or equal to MIN('ZDN',J+NY), either K is not equal to I or L is not equal to J, and ZDAT(K,L) is not a special value. (The values of NX and NY are as specified by the values of the parameters 'HLX' and 'HLY'.)

A similar definition is used for the position of a low.

2.21 Color-Setting Philosophy

A number of the CONPACK parameters specify the color of some object (like the informational label) or class of objects (like all contour lines for a given contour level). The default value of each such parameter is -1, which says that the color is to be determined by the current value of one of the GKS color indices. (The polyline color index is used for lines, the text color index for labels, and the fill area color index for filling label boxes.) If the value of the CONPACK color-setting parameter for a given object is given a value greater than or equal to zero, it specifies the color index of the color in which the object is to be drawn. Before any object is drawn, the values of the GKS color indices affected are saved; after the object is drawn, the saved values are restored.

This structure allows the use of a tiered approach to color setting. If no color setting whatsoever is done, contour plots are drawn entirely in the colors specified by the applicable default values of the GKS color indices. If, on the other hand, prior to calling CONPACK, one defines the color index “IC” (see the next section, “GKS Considerations”) and then uses

```
CALL GSPLCI (IC)
```

to change the GKS polyline color index, then all polylines drawn by CONPACK change color. Similarly, one may use the statement

```
CALL GSTXCI (IC)
```

to change the GKS text color index and the statement

```
CALL GSFACI (IC)
```

to change the GKS fill area color index; the first will cause labels drawn by CONPACK to change color and the second will cause label boxes filled by CONPACK to change color.

If, in addition or instead, CONPACK color-setting parameters are given values greater than or equal to zero, the objects or classes of objects to which those parameters apply are colored accordingly; these colors are used in preference to values preset by calls to GSPLCI, GSTXCI, or GSFACI.

A final opportunity to set color is provided by the user-supplied versions of the “change” routines, with names of the form CPCHxx; calls to GSPLCI, GSTXCI, and GSFACI may occur in such a routine and take precedence over color setting by any other means. Note that, if color is being set for drawing a label, then either or both of the polyline color index and the text color index may need to be set, depending on whether the labels are being drawn by calls to the GKS routine GPL (to draw polylines stroking out the characters) or by calls to the GKS routine GTX (to draw text). In particular, the routine PLCHHQ, in the package PLOTCHAR, which is called by CONPACK to draw labels, may be directed by the user to draw high-quality characters, which are stroked, medium-quality characters, which are also stroked, or low-quality characters, which are drawn by GTX.

2.22 CONPACK’s Grid Window

Note that the grid window for CONPACK is not the same as the grid window for AUTOGRAPH, since the former is expressed with respect to the plotter frame and the latter is expressed with respect to the AUTOGRAPH “graph window”. There is an easy conversion from one to the other, however. If AGFL, AGFR, AGFB, and AGFT specify the AUTOGRAPH graph window and AGDL, AGDR, AGDB, and AGDT specify the AUTOGRAPH grid window and CGDL, CGDR, CGDB, and CGDT specify the CONPACK grid window, then the following relationships ensure coincidence of the AUTOGRAPH and CONPACK grid windows:

```
CGDL=AGFL+AGDL*(AGFR-AGFL)
CGDR=AGFL+AGDR*(AGFR-AGFL)
CGDB=AGFB+AGDB*(AGFT-AGFB)
CGDT=AGFB+AGDT*(AGFT-AGFB)
```

Note what happens when AGFL and AGFB are 0’s and AGFR and AGFT are 1’s, which is the default situation; in that case CGDL=AGDL, CGDR=AGDR, CGDB=AGDB, and CGDT=AGDT.

2.23 GKS Considerations

Certain assumptions are made by CONPACK about the state of GKS, as follows:

(1) Like all the utilities in the NCAR graphics package, CONPACK assumes that GKS has been opened and that the desired workstations have been opened and activated. The statement

```
CALL OPNGKS
```

calls the SPPS routine OPNGKS, the GKS equivalent of which is

```
CALL GOPKS (6,0)
CALL GOPWK (1,2,1)
CALL GACWK (1)
```

creating a single metacode workstation associated with FORTRAN unit 2.

Similarly, at the end of one's program, the workstations must be deactivated and closed and then GKS must be closed. The statement

```
CALL CLSGKS
```

calls the SPPS routine CLSGKS, the GKS equivalent of which is

```
CALL GDAWK (1)
CALL GCLWK (1)
CALL GCLKS
```

(2) In versions of CONPACK prior to 3.1.3, it was more or less assumed that clipping was turned off. To turn clipping off yourself, use the statement

```
CALL GSCLIP (0)
```

If this is not done, and if you are using a pre-3.1.3 version of CONPACK to draw the informational label in its default position, which is outside the viewport, it will be clipped and fail to appear on the plot.

(3) It is assumed that the aspect source flags for various quantities are set to "individual". (The NCAR GKS package does this by default, but other packages may not.) To make sure that all the aspect source flags are set correctly, use the following code:

```
DIMENSION IASF(13)
...
DATA IASF / 13*1 /
...
CALL GSASF (IASF)
```

(4) Color fill of label boxes is done by CONPACK using calls to the GKS routine GFA; color fill of contour bands by the user will be done using similar calls. To get solid fill, rather than hollow fill, one must call a GKS routine to set the "fill area interior style":

```
CALL GSFAIS (1)
```

(This is because the default "fill area interior style", as mandated by the GKS standard, is "hollow", rather than "solid".)

(5) Color-setting by CONPACK is done by executing calls to the GKS routines GSPLCI, GSTXCI, and GSFACI, with user-defined color indices as arguments. The association of these color indices with colors on the workstations must have been defined previously by the user. This should be done by calling the GKS routine GSCR. The statement

```
CALL GSCR (IW,IC,RC,GC,BC)
```

defines, for workstation IW, color index IC, with RGB components RC, GC, and BC. To be consistent with the SPPS routines OPNGKS and CLSGKS, use IW = 1. The value of IC may be any non-negative integer. By default, color index 0 is associated with the color black, which is defined by (RC,GC,BC) = (0.,0.,0.) and is used as the background color, while color index 1 is associated with the color white, which is defined by (RC,GC,BC) = (1.,1.,1.).

3.0 SUBROUTINES

All of the CONPACK routines have six-character names beginning with the letters 'CP'. The user-callable ones are described in detail below.

3.1 CPBACK (ZDAT,RWRK,IWRK)

This routine draws a background for a contour plot.

The initial version of CPBACK does very little. User feedback will be useful in determining what this routine is eventually made to do.

3.1.1 Usage

The routine CPBACK may be called at any time after the initialization call to CPRECT, CPSPS1, or CPSPS2. If the parameter 'MAP' is zero, CPBACK draws the perimeter of the current viewport; it does this by calling PERIM, in the package GRIDAL, requesting 'ZDM'-1 major ticks on the horizontal sides of the perimeter and 'ZDN'-1 major ticks on the vertical sides. No minor ticks are drawn. If the parameter 'MAP' is non-zero, CPBACK does nothing.

3.1.2 Arguments

All three arguments are arrays used in a previous call to one of the routines CPRECT, CPSPS1, CPSPS2, CPMVRW, or CPMVIW, the contents of which must not have been changed by the user since that call.

ZDAT (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, or CPSPS2, input) is the data array.

RWRK (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVRW, input/output) is the current real workspace array.

IWRK (INTEGER array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVIW, input/output) is the current integer workspace array.

3.2 CPCHCF (IFLG)

This routine provide user control as a constant-field message is drawn.

3.2.1 Usage

The routine CPCHCF is not to be called by the user. It is called several times by CONPACK while a constant-field label is being drawn. The default version of CPCHCF does nothing. A user-supplied replacement may change attributes such as color and line width (by calling the SPPS routine SETUSV or the appropriate GKS routines). The text of the label being written may be retrieved by means of a “CALL CPGETC ('CTM',CVAL)”. The text of the label may be changed by means of a “CALL CPSETC ('CTM',CVAL)”; this should only be done during a call with IFLG = 1 or 3 and, if it is done for one of those two values, it should also be done for the other. When CPCHCF is called, the parameter 'ZDV' will have been set to the value of the field; its value may be retrieved and used by CPCHCF.

3.2.2 Arguments

IFLG (INTEGER, input) is positive if an action is about to be taken, negative if an action has just been completed. The action in question is defined by the absolute value of IFLG, as follows:

- The value 1 implies the determination of the size of the label (by means of a call to PLCHHQ, in the package PLOTCHAR, with ANGD=360.).
- The value 2 implies the filling of the box around the label, which is done before the drawing of the label itself.
- The value 3 implies the drawing of the label (by means of a call to PLCHHQ, in the package PLOTCHAR).
- The value 4 implies the drawing of the box around the label, which is done after the drawing of the label itself.

3.3 CPCHCL (IFLG)

This routine provides user control as contour lines are drawn.

3.3.1 Usage

The routine CPCHCL is not to be called by the user. It is called by the CONPACK routines CPCLDM and CPCLDR just before and just after the contour lines at each level are drawn. The default version of CPCHCL does nothing. A user-supplied replacement may change attributes such as color and line width (by calling the SPPS routine SETUSV or the appropriate GKS routines).

If the element of the parameter array 'CLU' corresponding to 'PAI' = -1 has been set non-zero to request the drawing of the edge of the grid, then CPCHCL will be called before and after that is done. Similarly, if the element of 'CLU' corresponding to 'PAI' = -2 has been set non-zero, then CPCHCL will be called before and after the drawing of the edges of the special-value areas, and, if the element of 'CLU' corresponding to 'PAI' = -3 has been set non-zero, then CPCHCL will be called before and after the drawing of the edges of the out-of-range areas.

When CPCHCL is called, the parameter 'PAI' will have been set to the index of the appropriate contour level (between 1 and 'NCL') or to one of the values -1, -2, or -3. By retrieving the value of 'PAI', CPCHCL can find out what line is being drawn; also, a CPGETx call to retrieve an element of a parameter array like 'CLD' will automatically get the correct one for the line being drawn.

3.3.2 Arguments

IFLG (INTEGER, input) is +1 if a line is about to be drawn, -1 if a line has just been drawn.

3.4 CPCHHL (IFLG)

This routine provides user control as high and low labels are drawn.

3.4.1 Usage

The routine CPCHHL is not to be called by the user. It is called several times by CPLBAM and/or CPLBDR while each high or low label is positioned and drawn. The default version of CPCHHL does nothing. A user-supplied replacement may change attributes such as color and line width (by calling the SPPS routine SETUSV or the appropriate GKS routines). The text of the label may be retrieved by means of a “CALL CPGETC ('CTM',CVAL)”. The text of the label may be changed by means of a “CALL CPSETC ('CTM',CVAL)”; this should only be done during a call with IFLG = 1, 3, 5, or 7; if it is done for one of the two values 1 and 3, it should also be done for the other; if it is done for one of the two values 5 and 7, it should also be done for the other. When CPCHHL is called, the parameter 'ZDV' will have been set to the value of the high or low being labelled; its value may be retrieved and used by CPCHHL.

3.4.2 Arguments

IFLG (INTEGER, input) is positive if an action is about to be taken, negative if an action has just been completed. The action in question is defined by the absolute value of IFLG, as follows:

- The value 1 implies the determination of the size of the label for a high (by means of a call to PLCHHQ, in the package PLOTCHAR, with ANGD = 360.).
- The value 2 implies the filling of the box around the label for a high, which is done before the drawing of the label itself.
- The value 3 implies the drawing of the label for a high (by means of a call to PLCHHQ, in the package PLOTCHAR).
- The value 4 implies the drawing of the box around the label for a high, which is done after the drawing of the label itself.
- The value 5 implies the determination of the size of the label for a low (by means of a call to PLCHHQ, in the package PLOTCHAR, with ANGD = 360.).
- The value 6 implies the filling of the box around the label for a low, which is done before the drawing of the label itself.
- The value 7 implies the drawing of the label for a low (by means of a call to PLCHHQ, in the package PLOTCHAR).
- The value 8 implies the drawing of the box around the label for a low, which is done after the drawing of the label itself.

3.5 CPCHIL (IFLG)

This routine provides user control as the informational label is drawn.

3.5.1 Usage

The routine CPCHIL is not to be called by the user. It is called several times by CPLBAM and/or CPLBDR while the informational label is positioned and drawn. The default version of CPCHIL does nothing. A user-supplied replacement may change attributes such as color and line width (by calling the SPPS routine SETUSV or the appropriate GKS routines). The text of the label may be retrieved by means of a “CALL CPGETC ('CTM',-CVAL)”. The text of the label may be changed by means of a “CALL CPSETC ('CTM',CVAL)”; this should only be done during a call with IFLG = 1 or IFLG = 3, and, if it is done for one of those values, it should be done for the other.

3.5.2 Arguments

IFLG (INTEGER, input) is positive if an action is about to be taken, negative if an action has just been completed. The action in question is defined by the absolute value of IFLG, as follows:

- The value 1 implies the determination of the size of the label (by means of a call to PLCHHQ, in the package PLOTCHAR, with ANGD=360.).
- The value 2 implies the filling of the box around the label, which is done before the drawing of the label itself.
- The value 3 implies the drawing of the label (by means of a call to PLCHHQ, in the package PLOTCHAR).
- The value 4 implies the drawing of the box around the label, which is done after the drawing of the label itself.

3.6 CPCHLL (IFLG)

This routine provides user control as contour line labels are drawn.

3.6.1 Usage

The routine CPCHLL is not to be called by the user. It is called several times by CPLBAM and/or CPLBDR while each contour-line label is positioned and drawn. The default version of CPCHLL does nothing. A user-supplied replacement may change attributes such as color and line width (by calling the SPPS routine SETUSV or the appropriate GKS routines). The text of the label may be retrieved by means of a “CALL CPGETC ('CTM',-CVAL)”. The text of the label may be changed by means of a “CALL CPSETC ('CTM',CVAL)”; this should only be done during a call with IFLG = 1 or 3 and, if it is done for one of those values, it should be done for the other.

When CPCHLL is called, the parameter 'PAI' will have been set to the index of the appropriate contour level. Thus, parameters associated with that level may easily be retrieved by calls to CPGETx.

3.6.2 Arguments

IFLG (INTEGER, input) is positive if an action is about to be taken, negative if an action has just been completed. The action in question is defined by the absolute value of IFLG, as follows:

- The value 1 implies the determination of the size of the label (by means of a call to PLCHHQ, in the package PLOTCHAR, with ANGD=360.).
- The value 2 implies the filling of the box around the label, which is done before the drawing of the label itself.

- The value 3 implies the drawing of the label (by means of a call to PLCHHQ, in the package PLOTCHAR).
- The value 4 implies the drawing of the box around the label, which is done after the drawing of the label itself.

3.7 CPCICA (ZDAT,RWRK,IWRK,ICRA,ICA1,ICAM,ICAN, ...)

(The remaining arguments are **XCPF**, **YCPF**, **XCQF**, and **YCQF**.)

This routine incorporates into a user's cell array color indices determined by examining where the user's contours lie relative to the cell array.

Special Note: If the internal parameter 'MAP' is non-zero, CPCICA requires that the routine CPMPXY be capable of doing the inverse transformation associated with that non-zero value of 'MAP'. As of release 3.1.3 of NCAR Graphics, the default version of CPMPXY does the inverses for the built-in mappings; users who supply their own versions of CPMPXY may need to upgrade them. See the description of the routine CPMPXY, below.

3.7.1 Usage

The routine CPCICA may be called at any time after the initialization call to CPRECT, CPSPS1, or CPSPS2. It is given an array of data to be contoured, a real workspace array, an integer workspace array, a GKS cell array, and dimensioning and positioning information for the cell array. For each cell in the cell array, CPCICA determines where on the plotter frame the center point of the cell falls and where that point is relative to the part of the plotter frame occupied by the user's data array (which may be just a simple rectangle or, if 'MAP' is non-zero, some more complicated shape). It is then possible to associate an area identifier (let's call it IAID) with the center point of the cell; there are four possibilities:

- If the center of the cell is inside the area occupied by the data grid, but not inside a special-value area, and the contents of the internal parameter arrays 'CLV', 'AIA', and 'AIB' imply that it is inside a particular contour band for which a non-zero area identifier is unambiguously specified, then IAID is given the value of that area identifier.
- If the center of the cell is inside the area occupied by the data grid, but not inside a special-value area, and the contents of the internal parameter arrays 'CLV', 'AIA', and 'AIB' do not unambiguously specify a particular area identifier, then IAID is given the value 0.
- If the center of the cell is inside the area occupied by the data grid and inside a special-value area (any box of the grid having the non-zero special value 'SPV' at one of its corners), then IAID is given the value of the internal parameter 'AIA' that is associated with the parameter array index 'PAI' = -2.

- If the center of the cell is outside the area occupied by the data grid, IAID will be given the value of the internal parameter 'AIA' that is associated with the parameter array index 'PAI' = -1.

Note that, to be sure of distinguishing all of the four possibilities from one another, it is necessary to give elements -1 and -2 of the parameter array 'AIA' values which are non-zero and different from the value of any other non-zero element of 'AIA' or 'AIB'.

Once an area identifier has been associated with a particular cell, that area identifier must be mapped into a color index that can be stored as the value of the corresponding element of the cell array. The internal parameter 'CAF' determines the way in which area identifiers are mapped into color indices, as follows:

- If 'CAF' is negative, the routine CPSCAE is called once for each cell. The default version of CPSCAE examines the value of the area identifier IAID, as defined above; if the value of IAID is greater than or equal to zero, it is used as the color index to be stored in the cell array; otherwise, no value is stored for that cell. A user version of CPSCAE may be supplied to do something more complicated; see the description of the routine CPSCAE, below.
- If 'CAF' is positive, the value 'CAF'+IAID, where IAID is the area identifier associated with the center point of the cell, is computed. If that value is non-negative, it is used as the color index to be stored in the cell array; otherwise, no value is stored for that cell. Note that, if element -1 of 'AIA' has the value -1, and if element -2 of 'AIA' has the value -2, and if, for I = 1 through 'NCL', element I of 'AIA' has the value I+1 and element I of 'AIB' has the value I, and if 'CAF' has the value 3, then the color indices stored in the cell array will have values from 1 through 'NCL'+4, where the value 1 implies a special-value area, the value 2 implies an area outside the grid, the value 3 implies an area inside the grid for which no color index has been unambiguously specified (which should not occur, in this case) and each of the values 4 through 'NCL'+4 implies a particular contour band.

The default value of 'CAF' is zero, so that CPSCAE is not called: if IAID is non-negative, it is used as the desired color index for a particular cell; otherwise, the color index for the cell in the cell array is unchanged.

Note that, if cells lying outside the data grid or in special-value areas of the data grid are not to have their color indices reset, elements -1 and -2 in the parameter array 'AIA' should be made large and negative; if, on the other hand, it is desired that such cells should be marked, they must be given values which will result in the generation (by whichever scheme is selected by the value of 'CAF') of color indices greater than or equal to zero.

3.7.2 Arguments

The first three arguments are arrays used in a previous call to one of the routines CPRECT, CPSPS1, CPSPS2, CPMVRW, or CPMVIW, the contents of which must not have been changed by the user since that call.

ZDAT (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, or CPSPS2, input) is the data array.

RWRK (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVRW, input/output) is the current real workspace array.

IWRK (INTEGER array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVIW, input/output) is the current integer workspace array.

ICRA (INTEGER array, dimensioned ICA1 by “n”, where “n” is greater than or equal to the value of the argument ICAN, input/output) is the user’s cell array.

ICA1 (INTEGER, input) is the first dimension of the FORTRAN array ICRA, which contains the user’s cell array.

ICAM (INTEGER, input) is the first dimension of the user’s cell array.

ICAN (INTEGER, input) is the second dimension of the user’s cell array.

XCPF and **YCPF** (REAL, input) are the coordinates, in the fractional coordinate system, of a point P. P is the point at that corner of the rectangular area into which the cell array maps that corresponds to the cell (1,1).

XCQF and **YCQF** (REAL, input) are the coordinates, in the fractional coordinate system, of a point Q. Q is the point at that corner of the rectangular area into which the cell array maps that corresponds to the cell (ICAM,ICAN).

3.8 CPCLAM (ZDAT,RWRK,IWRK,IAMA)

This routine adds contour lines to an area map. This is part of the process of drawing a solid-fill contour plot. The example “cpex13” shows what this routine puts into an area map.

3.8.1 Usage

The routine CPCLAM may be called at any time after the initialization call to CPRECT, CPSPS1, or CPSPS2 to add contour lines generated from the data in the array ZDAT to the area map in the array IAMA. The area map must previously have been initialized by a call to the routine ARINAM, in the package AREAS.

The contour lines added to the area map are as specified by the first 'NCL' elements of the parameter arrays 'CLV', 'AIA', and 'AIB'. If 'NCL' is zero, CPPKCL is called to generate these values. The contour levels defined by the first ’NCL’ elements of the parameter array ’CLV’ are then examined. If a given contour level is associated with a non-zero value of ’AIA’ and/or a non-zero value of ’AIB’, the contour lines at that contour level are added to the area map. If there is an associated non-zero value of ’AIA’, it is used as the area identifier for the area “above” the line (where field values are greater than they are along the line); otherwise, a zero is used. If there is an associated non-zero value of ’AIB’, it is used as the area identifier for the area “below” the line (where field values are less than they are along the line); otherwise, a zero is used. Note that a given contour level may occur more than once in the internal parameter array ’CLV’, but there must be at most one non-zero value of ’AIA’ and at most one non-zero value of ’AIB’ associated with it; otherwise, an error exit occurs.

Other types of lines are added to the area map by CPCLAM: the edge of the current viewport and possibly a set of vertical lines within the viewport; the edge of the grid; the edges of special-value areas, if any; the edges of out-of-range areas, if any. The area identifier for the outside of the viewport is always -1. Elements of the parameter array 'AIA' for 'PAI' = -1, -2, and -3 may be used to specify the area identifiers to be used for the outside of the grid, the inside of a special-value area, and the inside of an out-of-range area, respectively; the default values of all three are -1’s. (As of version 3.1.3, the first of these has been changed to “0”.) Area identifiers for all other sides of these edges are determined from the area-identifier information given for the contour levels.

Lines are added to the area map in the following order: the edge of the viewport and the vertical lines within it, edges of the out-of-range areas, if any; the edge of the grid; edges of the special-value areas, if any; the contour lines, in order of increasing contour level.

The edge of the viewport may actually be added to the area map twice - once as part of the edge group with group identifier 'GIC', and again as part of the edge group with group identifier 'GIS'. The object of the former edge group is to prevent problems which arise when mapping is turned on and the mapping function has a line of discontinuity (as, for example, when using EZMAP with a cylindrical equidistant projection). The object of the latter edge group is to break up the areas represented by the area map into smaller pieces; whether this is done or not is under control of the user, by means of the parameters 'NVS' and 'GIS'; for more information, see the descriptions of those parameters.

If, during the last call to CPRECT, CPSPS1, or CPSPS2, the data being contoured were found to be essentially constant, then no contour lines are added to the area map; the other lines are added, however.

3.8.2 Arguments

The first three arguments are arrays used in a previous call to one of the routines CPRECT, CPSPS1, CPSPS2, CPMVRW, or CPMVIW, the contents of which must not have been changed by the user since that call.

ZDAT (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, or CPSPS2, input) is the data array.

RWRK (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVRW, input/output) is the current real workspace array.

IWRK (INTEGER array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVIW, input/output) is the current integer workspace array.

IAMA (INTEGER array, dimensioned as specified in a call to ARINAM, in the package AREAS) is the array containing the area map to which contour lines are to be added.

3.9 CPCLDM (ZDAT,RWRK,IWRK,IAMA,RTPL)

This routine draws contour lines masked by an existing area map. The object of this may be simply to avoid drawing contour lines through label boxes, but the routine may be used for more complicated tasks, like limiting the drawing of contour lines to the ocean areas on an EZMAP background.

3.9.1 Usage

The routine CPCLDM may be called at any time after the initialization call to CPRECT, CPSPS1, or CPSPS2 to draw contour lines masked by an existing area map. Actually, CPCLDM does not draw the lines; it generates them, masks them against a user-specified area map, and generates a set of calls to a user-specified routine (one call for each polyline resulting from the masking process). Each such polyline lies entirely within precisely one of the areas defined by the area map. The user routine may use the information provided by its arguments, describing the area the polyline is in, to decide whether or not to draw the polyline.

The contour lines generated are those specified by the first 'NCL' elements of the parameter arrays 'CLV' and 'CLU'. If 'NCL' is zero, CPPKCL is called to generate these values. Each element of 'CLV' specifies a contour level and the corresponding element of 'CLU' specifies whether or not contour lines are to be generated at that level. If the parameter 'T2D' has a non-zero value, the contour lines are smoothed, using cubic splines under tension.

If the element of the parameter array 'CLU' corresponding to 'PAI' = -1 is non-zero, the edge of the grid is also generated. If the element of 'CLU' corresponding to 'PAI' = -2 is non-zero, the edges of special-value areas (if any) are generated. If the element of 'CLU' corresponding to 'PAI' = -3 is non-zero, the edges of out-of-range areas (if any) are generated. The default values are such that none of these edges is generated.

Groups of lines are generated in the following order: contour lines for each of the specified levels, in ascending numeric order; the edges of special-value areas, if any; the edges of out-of-range areas, if any; the edge of the grid.

The color, dash pattern, and line width to be used for the lines drawn may be specified by elements of the parameter arrays 'CLC', 'CLD', and 'CLL', respectively. Each of these contains elements corresponding to values of 'PAI' from 1 to 'NCL' and three special elements, corresponding to 'PAI' = -1, -2, and -3. Before and after each group of lines is generated, the routine CPCHCL is called; a user-supplied version of this routine may override the settings of color, dash pattern, and line width. (Also, of course, the user routine which actually does the drawing may override the settings of these quantities.)

If the dash-pattern-usage parameter 'DPU' is zero, it is assumed that lines are to be drawn by calling the SPPS routine CURVE; specified dash patterns are not used. If 'DPU' is greater than zero, it is assumed that lines are to be drawn by calling the DASHCHAR routine CURVED, while if 'DPU' is less than zero, it is assumed that lines are to be drawn by calling the DASHPACK routine DPCURV; in either case, specified dash patterns are used. If 'DPU' is non-zero and ABS('LLP') = 1, then, for those lines which are to be labelled, the dash pattern is constructed by replicating the dash pattern specified by the appropriate element of 'CLD' 'DPU' times and then adding to that the numeric label specified by the appropriate element of 'LLT'.

If, during the last call to CPRECT, CPSPS1, or CPSPS2, the data being contoured were found to be essentially constant, then no contour lines are generated; instead, the constant-field label is written. Other lines are still generated.

3.9.2 Arguments

The first three arguments are arrays used in a previous call to one of the routines CPRECT, CPSPS1, CPSPS2, CPMVRW, or CPMVIW, the contents of which must not have been changed by the user since that call.

ZDAT (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, or CPSPS2, input) is the data array.

RWRK (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVRW, input/output) is the current real workspace array.

IWRK (INTEGER array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVIW, input/output) is the current integer workspace array.

IAMA (INTEGER array, dimensioned as specified in a call to ARINAM, in the package AREAS, input/output) is the array containing the area map which is to be used to mask the contour lines as they are drawn.

RTPL (EXTERNAL subroutine) is the user subroutine which is to process the polylines which result from masking the generated contour lines and other edges against the area map. It must be declared EXTERNAL in the routine which calls CPCLDM. It will be called repeatedly and must have the following form:

```
SUBROUTINE RTPL (XCS,YCS,NCS,IAI,IAG,NAI)
  DIMENSION XCS(*),YCS(*),IAI(*),IAG(*)
  ...
  (CODE TO PROCESS POLYLINE DEFINED BY ARGUMENTS)
  ...
  RETURN
END
```

The real arrays XCS and YCS hold the X and Y coordinates of NCS points defining a polyline which is to be considered for drawing. For each I greater than or equal to 1 and less than or equal to NAI, IAI(I) is the area identifier for the area in which the polyline lies, relative to the area-identifier group IAG(I). The X and Y coordinates are all normalized device coordinates and it may be assumed that the appropriate SET call has been done. If it is decided to draw the line, it may be done with a call to the SPPS routine CURVE, to the DASHCHAR routine CURVED, to the DASHPACK routine DPCURV, or to the GKS routine GPL. The value of 'PAI' will have been set to the appropriate value (1 through 'NCL', -1, -2, or -3) for the line of which the polyline is a part. The color and line width will have been set as implied by the values of the appropriate elements of the parameter arrays 'CLC' and 'CLL'. The dash pattern will have been defined as implied by the value of the appropriate elements of the parameter arrays 'CLD' and 'CLU' and the value of the parameter 'DPU'. If a dash pattern is defined, it may be retrieved by a “CALL CPGETC ('CTM',CVAL)”.

If the only object of using CPCLDM is to avoid drawing contour lines through label boxes, then the CONPACK routine CPDRPL may be used for RTPL. In the routine that calls CPCLDM, insert the declaration

```
EXTERNAL CPDRPL
```

and then use CPDRPL for the last argument.

For more information, see the description of the argument LPR of the subroutine ARDRLN, in the documentation of the package AREAS.

3.10 CPCLDR (ZDAT,RWRK,IWRK)

This routine draws contour lines.

3.10.1 Usage

The routine CPCLDR may be called at any time after the initialization call to CPRECT, CPSPS1, or CPSPS2 to draw contour lines.

The contour lines drawn are those specified by the first 'NCL' elements of the parameter arrays 'CLV' and 'CLU'. If 'NCL' is zero, CPPKCL is called to generate these values. Each element of 'CLV' specifies a contour level and the corresponding element of 'CLU' specifies whether or not contour lines are to be drawn at that level and whether or not the lines are to be labelled. If the parameter 'T2D' has a non-zero value, the contour lines are smoothed, using cubic splines under tension.

If the element of the parameter array 'CLU' corresponding to 'PAI' = -1 is non-zero, the edge of the grid is also drawn. If the element of 'CLU' corresponding to 'PAI' = -2 is non-zero, the edges of special-value areas (if any) are drawn. If the element of 'CLU' corresponding to 'PAI' = -3 is non-zero, the edges of out-of-range areas (if any) are drawn. The default values are such that none of these edges are drawn.

Groups of lines are drawn in the following order: contour lines for each of the specified levels, in ascending numeric order; the edges of special-value areas, if any; the edges of out-of-range areas, if any; the edge of the grid.

The color, dash pattern, and line width to be used for the lines drawn may be specified by elements of the parameter arrays 'CLC', 'CLD', and 'CLL', respectively. Each of these contains elements corresponding to values of 'PAI' from 1 to 'NCL' and three special elements, corresponding to 'PAI' = -1, -2, and -3. Before and after each group of lines is drawn, the routine CPCHCL is called; a user-supplied version of this routine may override the settings of color, dash pattern, and line width.

If the dash-pattern-usage parameter 'DPU' is zero, lines are drawn by calling the SPPS routine CURVE; they are all solid and unlabelled. If 'DPU' is greater than zero, lines are drawn by calling the DASHCHAR routine CURVED, while if 'DPU' is less than zero, lines are drawn by calling the DASHPACK routine DPCURV; in either case, they are solid or dashed, depending on the dash pattern specified by the appropriate element of 'CLD'. If 'DPU' is non-zero and ABS('LLP') = 1, the dash pattern for those lines which are to be labelled is constructed by replicating the dash pattern specified by the appropriate element of 'CLD' 'DPU' times and then adding to it the characters specified by the apppropriate element of 'LLT'. If 'DPU' is non-zero and ABS('LLP') is greater than 1, the lines drawn will pass through any labels drawn by CPLBDR; if this is undesirable, CPLBAM can be used to put the label boxes into an area map and CPCLDM can be used to draw only those portions of the contour lines which do not lie inside the label boxes.

If, during the last call to CPRECT, CPSPS1, or CPSPS2, the data being contoured were found to be essentially constant, then no contour lines are drawn; instead, the constant-field label is written. Other lines are still drawn.

3.10.2 Arguments

All three arguments are arrays used in a previous call to one of the routines CPRECT, CPSPS1, CPSPS2, CPMVRW, or CPMVIW, the contents of which must not have been changed by the user since that call.

ZDAT (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, or CPSPS2, input) is the data array.

RWRK (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVRW, input/output) is the current real workspace array.

IWRK (INTEGER array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVIW, input/output) is the current integer workspace array.

3.11 CPCLTR (ZDAT,RWRK,IWRK,CLVL,IJMP, ...)

(The remaining arguments are **IRW1**, **IRW2**, and **NRWK**.)

This routine traces the contour lines at a given level and retrieves them for some sort of user-defined processing. The contour lines are traced in such a way that lower field values (less than CLVL) are to the left and higher field values (greater than CLVL) are to the right (where “left” and “right” are defined from the standpoint of someone standing at one point of the contour line, looking toward the next).

3.11.1 Usage

The routine CPCLTR may be called at any time after the initialization call to CPRECT, CPSPS1, or CPSPS2. It is called using code like the following:

```
      IJMP=0
  101 CALL CPCLTR (ZDAT,RWRK,IWRK,CLEV,IJMP,IRW1,IRW2,NRWK)
      IF (IJMP.NE.0) THEN
        (PROCESS SEGMENT OF CONTOUR LINE THAT HAS BEEN PUT IN RWRK.)
        GO TO 101
      END IF
```

CPCLTR is called repeatedly. Initially, IJMP is zeroed. Upon each return from CPCLTR, if the value of IJMP is non-zero, X and Y coordinates have been placed in the real work-space array RWRK and the values of IRW1, IRW2, and NRWK have been set to indicate where the coordinates were put. After the coordinates have been processed, CPCLTR is called again. If, upon return from CPCLTR, the value of IJMP is zero, all contour lines at the specified level have been processed.

If the parameter 'T2D' has a non-zero value, the contour lines are smoothed, using cubic splines under tension.

3.11.2 Arguments

The first three arguments are arrays used in a previous call to one of the routines CPRECT, CPSPS1, CPSPS2, CPMVRW, or CPMVIW, the contents of which must not have been changed by the user since that call.

ZDAT (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, or CPSPS2, input) is the data array.

RWRK (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVRW, input/output) is the current real workspace array.

IWRK (INTEGER array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVIW, input/output) is the current integer workspace array.

CLEV (REAL, input) is the desired contour level.

IJMP (INTEGER, input/output) is initially set to zero by the user. Upon return from CPCLTR, it will be non-zero if coordinates of a contour line segment have been placed in RWRK; it will be zero if all contour line segments at the given level have been generated.

IRW1 (INTEGER, output) is a pointer to the place in RWRK where X coordinates have been stored. They will be in RWRK(IRW1+I) for I = 1, 2, ..., NRWK.

IRW2 (INTEGER, output) is a pointer to the place in RWRK where Y coordinates have been stored. They will be in RWRK(IRW2+I) for I = 1, 2, ..., NRWK.

NRWK (INTEGER, output) is the number of points on the contour line segment. The points comprising the segment will be (RWRK(IRW1+I),RWRK(IRW2+I)) for I = 1, 2, ... NRWK.

3.12 CPCNRC (ZDAT,KZDT,MZDT,NZDT, ...)

(The remaining arguments are **FLOW**, **FHGH**, **FINC**, **NSET**, **NHGH**, and **NDSH**.)

This routine simulates the behavior of the old routine CONREC; it has the same arguments and will produce similar output.

3.12.1 Usage

Use the FORTRAN statement

```
CALL CPCNRC (ZDAT,KZDT,MZDT,NZDT,FLOW,FHGH,FINC,NSET,NHGH,NDSH)
```

to produce a complete contour plot. The appearance of the plot produced may vary, depending on the setting of internal parameters of CONPACK. The following should be noted:

- (1) By default, contour lines will be labelled using the old CONREC scheme of appending the label for a line to the dash pattern for the line. To use one of the new schemes, place a “CALL CPSETI ('LLP - LINE LABEL POSITIONING',n)”, where “n” has the value “2” or “3”, before the call to CPCNRC. CPCNRC retrieves the value of 'LLP' and, if the value is a 2 or a 3, modifies the calls that it does so as to properly shield labels from having contour lines drawn through them.
- (2) By default, high-quality characters will be used in the informational label, in high/low labels, in point-value labels, and in contour-line labels written when 'LLP' is 2 or 3. To use lower-quality characters, place a “CALL PCSETI ('QUALITY',1)” or a “CALL PCSETI ('QUALITY',2)” prior to the call to CPCNRC. (Note that the “PC” is no misprint; the routine being called is in the package PLOTCHAR, which is used by CONPACK to plot characters. See the documentation of PLOTCHAR for more information.)

- (3) If the output from CPCNRC is to be overlaid on an EZMAP background, longitude is a linear function of the first subscript of the data array, and latitude is a linear function of the second subscript of the data array, one can simply call EZMAP to draw the background, set the parameter 'MAP' to 1, set the parameters 'XC1', 'XCM', 'YC1', and 'YCN' to specify the longitude and latitude ranges, and then call CPCNRC to draw the contour plot. Note that 'XCM' must be numerically larger than 'XC1'. If, for example, your data run from 175 degrees east of Greenwich at the left edge to 160 degrees west of Greenwich at the right edge (which is -160 degrees east), then you should set 'XC1' = 175 and 'XCM' = 200 to achieve the proper mapping.
- (4) To smooth the contour lines in the same way that they would have been smoothed by the "smooth" or "super" versions of CONREC, just insert a "CALL CPSETR ('T2D - TENSION ON THE 2D SPLINES',t)", where "t" is the desired tension, to turn on the smoother.
- (5) By default, no scale factor is used. (Because CONPACK can write contour line labels in scientific notation, it was considered less important to use a scale factor.) If it is desired that a scale factor should be used, the value of the parameter 'SFS' should be set prior to calling CPCNRC.
- (6) CPCNRC contains a real workspace array, an integer workspace array, and an area map (used to prevent contour lines from passing through labels when 'LLP' has the value 2 or 3). The lengths of these arrays are 5000, 2000, and 12000, respectively; if this proves insufficient, you must obtain a copy of the code for CPCNRC, adjust the dimensions as required, and compile it in place of the default version.
- (7) Certain parameters are used by CPCNRC; user calls to reset them prior to the call to CPCNRC will therefore have no effect. These parameters are as follows: 'AIA', 'AIB', 'CIS', 'CIU', 'CLC', 'CLD', 'CLL', 'CLS', 'CLV', 'CLU', 'CMN', 'CMX', 'HLT' (and therefore 'HIT' and 'LOT'), 'LLC', 'LLT', 'NCL', 'SET', 'VPB', 'VPL', 'VPR', 'VPS', and 'VPT'.
- (8) The user callback routines with names of the form CPCHxx will be of more value when used in conjunction with CPCNRC than when used in conjunction with the more basic routines in the package. For example, the only way to change the color of the zero contour line drawn by a call to CPCNRC is to supply one's own version of the routine CPCHCL; there is no way, short of modifying the code of CPCNRC, to use the parameter which controls the color of the zero contour line.
- (9) Like CONREC, CPCNRC does no call to FRAME. To advance the frame, put a "CALL FRAME" after the call to CPCNRC.

3.12.2 Arguments

ZDAT (REAL array, dimensioned KZDT x n, where "n" is greater than or equal to NZDT, input) is the array containing the data to be contoured.

KZDT (INTEGER, input) is the first dimension of the FORTRAN array ZDAT.

MZDT (INTEGER, input) is the first dimension of the array of data in ZDAT. MZDT must be less than or equal to KZDT.

NZDT (INTEGER, input) is the second dimension of the array of data in ZDAT. NZDT must be less than or equal to the declared second dimension of the array ZDAT.

FLOW (REAL, input) is the desired lowest contour level. If FLOW is equal to or greater than FHGH, CONPACK will choose the set of contour levels.

FHGH (REAL, input) is the desired highest contour level. If FHGH is equal to or less than FLOW, CONPACK will choose the set of contour levels.

FINC (REAL, input) says how contour levels are to be chosen. There are two possibilities:

- If FINC is greater than zero, it specifies the desired contour interval to be used. In this case, if FLOW is less than FHGH, the intervals used will be FLOW, FLOW+FINC, FLOW+2*FINC, ... FLOW+n*FINC, where "n" is the largest integer such that FLOW+n*FINC is less than or equal to FHGH. If FLOW is greater than or equal to FHGH, the contour levels will be those integer multiples of FINC which fall between the minimum value in ZDAT and the maximum value in ZDAT.
- If FINC is less than or equal to zero, CONPACK will choose the contour interval in such a way as to give at least 16 contour levels (if FINC is zero) or MAX(1,INT(-FINC)) contour levels (if FINC is less than zero) between the minimum and maximum values in ZDAT. All the contour levels will be integer multiples of the chosen interval. If FLOW is less than FHGH, no contour lines will be drawn for chosen contour levels which are outside the range (FLOW,FHGH).

NSET (INTEGER, input) says how the contour plot is to be mapped onto the plotter frame. There are three basic possibilities:

- If the value of NSET, modulo three, is zero, the "standard configuration" will be used. CONPACK will be directed to call SET. The portion of the plotter frame bounded by the lines "X=.05", "X=.95", "Y=.05", and "Y=.95" (in the "fractional" or "normalized-device-coordinate" system) will be used. The shape of the plot will be determined by the values of the internal parameters 'WDL', 'WDR', 'WDB', and 'WDT'; by default, the ratio of the plot's width to its height will be MZDT/NZDT. If the ratio of the width to the height is less than 1/4 or greater than 4, the plot will be made square. CPBACK will be called to draw a perimeter.
- If the value of NSET, modulo three, is equivalent to minus one, the contour plot will fill the current viewport. CONPACK will be directed to call SET. The portion of the plotter frame used will be that bounded by the lines "X=xl", "X=xr", "Y=yb", and "Y=yt", where "xl", "xr", "yl", and "yr" are obtained by means of a

  ```
  CALL GETSET (xl,xr,yl,yr,...)
  ```

  The plot will fill this entire area. CPBACK will not be called to draw a perimeter.
- If the value of NSET, modulo three, is equivalent to plus one, CONPACK will be directed not to call SET. It will be assumed that the user has done the appropriate call. CPBACK will not be called to draw a perimeter. Use this option when overlaying CPCNRC output on an existing background (one drawn by EZMAP, for example).

If the value of NSET is -1, 0, or 1, CPCNRC will, prior to returning control to the caller, restore the SET call to what it was on entry (which is how CONREC works). If the value of NSET is less than -1 or greater than 1, CPCNRC will not so restore the SET call; this may be useful if other objects are to be drawn on the plot produced by CPCNRC.

NHGH (INTEGER, input) says whether highs and lows or data points are to be labelled. There are three possibilities:

- If NHGH is zero, each high is marked with an "H" and each low is marked with an "L"; the value is written as a subscript of the "H" or "L".
- If NHGH is greater than zero, each data point is marked with the value at that point. No attempt is made to cull overlapping values, so using this option when there are too many data points may result in a mess. The values of the CONPACK parameters 'HLA' and 'HLS' are retrieved and used to determine the angle at which the values are written and the size of the characters used. They may be used to reduce the clutter somewhat.
- If NHGH is less than zero, neither of the above is done.

NDSH (INTEGER, input) may be used to specify a 10-bit dash pattern to be used. (If ABS(NDSH) is 0, 1, or 1023, all solid lines are used.) If NDSH is greater than zero, the specified dash pattern is used for all contour lines; if NDSH is less than zero, the dash pattern is used only for negative-valued contour lines. Note: the 10-bit pattern is actually made into a 16-bit pattern by appending a copy of the high-order six bits.

3.13 CPDRPL (XCS,YCS,NCS,IAI,IAG,NAI)

This routine provides a useful polyline-drawer for the routine CPCLDM (the description of which appears earlier in this section).

3.13.1 Usage

If CPCLDM is called, and the only object of using it, instead of CPCLDR, to draw contour lines is to avoid drawing the lines through labels, then, in the routine which calls CPCLDM, put the declaration

```
EXTERNAL CPDRPL
```

and, in the call to CPCLDM, use CPDRPL for the argument RTPL. Each time CPDRPL is called, it draws the polyline defined by its first three arguments if, and only if, none of the area identifiers defined by the other three arguments is negative.

3.13.2 Arguments

XCS (a REAL array of dimension at least NCS, input) is an array containing the X coordinates of NCS points defining a polyline.

YCS (a REAL array of dimension at least NCS, input) is an array containing the Y coordinates of NCS points defining a polyline.

NCS (INTEGER, input) is the number of points defining the polyline.

IAI (an INTEGER array of dimension at least NAI, input) is an array of area identifiers for the area in which the polyline lies. For each I from 1 to NAI, IAI(I) is the area identifier of the area with respect to the edge group IAG(I).

IAG (an INTEGER array of dimension at least NAI, input) is an array of group identifiers. See the description of IAI, above.

NAI (INTEGER, input) is the number of area identifiers in the array IAI and the number of group identifiers in the array IAG.

3.14 CPEZCT (ZDAT,MZDT,NZDT)

This routine simulates the behavior of the old routine EZCNTR; it has the same arguments and will produce similar output.

3.14.1 Usage

Use the statement

```
CALL CPEZCT (ZDAT,MZDT,NZDT)
```

to produce a complete contour plot. The effect will be exactly the same as if one has executed the statements

```
CALL CPCNRC (ZDAT,MZDT,MZDT,NZDT,0.,0.,0.,0,0,-682)
CALL FRAME
```

See the description of CPCNRC, above.

3.14.2 Arguments

ZDAT (REAL array, dimensioned MZDT x NZDT, input) is the array containing the data to be contoured.

MZDT (INTEGER, input) is the first dimension of the FORTRAN array ZDAT and of the array of data stored in it.

NZDT (INTEGER, input) is the second dimension of the FORTRAN array ZDAT and of the array of data stored in it.

3.15 CPGETC (PNAM,CVAL)

This routine is used to get the current value of a parameter of type CHARACTER.

3.15.1 Usage

Use the statement

```
CALL CPGETC (PNAM,CVAL)
```

at any time to retrieve in CVAL the current character value of the parameter whose name is PNAM. If that parameter is an array, the array element specified by the current value of the parameter 'PAI' will be the one retrieved.

3.15.2 Arguments

PNAM (CHARACTER, input) is the name of a parameter whose character value is to be retrieved. Only the first three characters of PNAM are examined. It is recommended that the rest of the character string be used to improve the readability of the code. For example, instead of just 'LLT', use 'LLT - LINE LABEL TEXT'.

CVAL (CHARACTER, output) is a variable in which the value of the parameter specified by PNAM is to be returned.

3.16 CPGETI (PNAM,IVAL)

This routine is used to get the current integer value of a parameter.

3.16.1 Usage

Use the statement

```
CALL CPGETI (PNAM,IVAL)
```

at any time to retrieve in IVAL the current integer value of the parameter whose name is PNAM. If that parameter is an array, the array element specified by the current value of the parameter 'PAI' will be the one retrieved.

3.16.2 Arguments

PNAM (CHARACTER, input) is the name of a parameter whose integer value is to be retrieved. Only the first three characters of PNAM are examined. It is recommended that the rest of the character string be used to improve the readability of the code. For example, instead of just 'LLP', use 'LLP - LINE LABEL POSITIONING'.

IVAL (INTEGER, output) is a variable in which the value of the parameter specified by PNAM is to be returned. If the internal parameter is of type INTEGER and has the value “i”, then IVAL = i; if the internal parameter is of type REAL and has the value “r”, then IVAL = INT(r).

3.17 CPGETR (PNAM,RVAL)

This routine is used to get the current real value of a parameter.

3.17.1 Usage

Use the statement

```
CALL CPGETR (PNAM,RVAL)
```

at any time to retrieve in RVAL the current real value of the parameter whose name is PNAM. If that parameter is an array, the array element specified by the current value of the parameter 'PAI' will be the one retrieved.

3.17.2 Arguments

PNAM (CHARACTER, input) is the name of a parameter whose real value is to be retrieved. Only the first three characters of PNAM are examined. It is recommended that the rest of the character string be used to improve the readability of the code. For example, instead of just 'SFS', use 'SFS - SCALE FACTOR SELECTOR'.

RVAL (REAL, output) is a variable in which the value of the parameter specified by PNAM is to be returned. If the internal parameter is of type INTEGER and has the value “i”, then RVAL = REAL(i); if the internal parameter is of type REAL and has the value “r”, then RVAL = r.

3.18 CPLBAM (ZDAT,RWRK,IWRK,IAMA)

This routine is used to add label boxes (for the informational label, high and low labels, and contour-line labels) to an area map. The ultimate object of this will usually be to prevent contour lines drawn by CPCLDM from passing through labels or to prevent fill of the label boxes as contour bands are filled.

3.18.1 Usage

The routine CPLBAM may be called at any time after the initialization call to CPRECT, CPSPS1, or CPSPS2 to add label boxes to an area map. If, during the last call to CPRECT, CPSPS1, or CPSPS2, the data being contoured were found to be essentially constant, then the box for the constant-field label is added to the area map. Otherwise, boxes for the informational label, the high/low labels, and/or the contour-line labels are added.

A box for the informational label is added only if the parameter 'ILT', which specifies the text of that label, is non-blank.

Boxes for the high labels are added only if the parameter 'HIT', which specifies the text of those labels, is non-blank.

Boxes for the low labels are added only if the parameter 'LOT', which specifies the text of those labels, is non-blank.

Boxes for the contour line labels are added only if the parameter 'LLP', which says how those labels are to be positioned, has an absolute value of 2 or 3, and if, for some I between 1 and 'NCL', inclusive, the Ith element of the parameter array 'CLU' has a value implying that contour lines at the contour level specified by the Ith element of 'CLV' are to be labelled.

3.18.2 Arguments

The first three arguments are arrays used in a previous call to one of the routines CPRECT, CPSPS1, CPSPS2, CPMVRW, or CPMVIW, the contents of which must not have been changed by the user since that call.

ZDAT (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, or CPSPS2, input) is the data array.

RWRK (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVRW, input/output) is the current real workspace array.

IWRK (INTEGER array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVIW, input/output) is the current integer workspace array.

IAMA (INTEGER array, dimensioned as specified in a call to ARINAM) is the array holding the area map to which the label boxes are to be added.

3.19 CPLBDR (ZDAT,RWRK,IWRK)

This routine draws labels (an informational label, high and low labels, and line labels).

3.19.1 Usage

The routine CPLBDR may be called at any time after the initialization call to CPRECT, CPSPS1, or CPSPS2 to draw labels. If, during the last call to CPRECT, CPSPS1, or CPSPS2, the data being contoured were found to be essentially constant, then the constant-field label is drawn. Otherwise, the informational label, the high/low labels, and/or the contour-line labels are drawn.

The informational label is drawn only if the parameter 'ILT', which specifies the text of that label, is non-blank.

High labels are drawn only if the parameter 'HIT', which specifies the text of those labels, is non-blank.

Low labels are drawn only if the parameter 'LOT', which specifies the text of those labels, is non-blank.

Contour line labels are drawn only if the parameter 'LLP', which says how those labels are to be positioned, has an absolute value of 2 or 3, and if, for some I between 1 and 'NCL', inclusive, the Ith element of the parameter array 'CLU' has a value implying that contour lines at the contour level specified by the Ith element of 'CLV' are to be labelled.

3.19.2 Arguments

All three arguments are arrays used in a previous call to one of the routines CPRECT, CPSPS1, CPSPS2, CPMVRW, or CPMVIW, the contents of which must not have been changed by the user since that call.

ZDAT (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, or CPSPS2, input) is the data array.

RWRK (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVRW, input/output) is the current real workspace array.

IWRK (INTEGER array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVIW, input/output) is the current integer workspace array.

3.20 CPMPXY (IMAP,XINP,YINP,XOTP,YOTP)

This routine maps CONPACK output from a rectangular coordinate system superimposed on the data grid to some other coordinate system.

Special Note: As of version 3.1.3 of NCAR Graphics, the default version of CPMPXY has been upgraded to do inverse mappings from whatever the current user coordinate system is to the rectangular coordinate system superimposed on the data grid. This has made it possible to add to CONPACK the new routine CPCICA, which adds color indices to a user's cell array, and to improve the way in which the routine CPCLAM generates the edges separating areas which are visible under the current mapping from areas which are invisible under that mapping. The default version of CPMPXY has also been changed to make it possible for a CONPACK routine using CPMPXY to ask it whether or not it is capable of doing a particular mapping. Users who supply their own versions of CPMPXY and wish to use CPCICA or to take advantage of the improvements in CPCLAM must upgrade CPMPXY in the same way that the default version has been upgraded.

3.20.1 Usage

CPMPXY is not to be called by the user. It is called by CONPACK when the parameter 'MAP' is non-zero. Each call is intended 1) to inquire whether a given mapping is defined by CPMPXY, or 2) to map the X and Y coordinates of a single point, whose position is known relative to the data grid, to X and Y coordinates in some other coordinate system or 3) (as of version 3.1.3) to do the inverse mapping. The default version of CPMPXY is as follows:

```
SUBROUTINE CPMPXY (IMAP,XINP,YINP,XOTP,YOTP)
  IF (IMAP.EQ.0) THEN
    IF (INT(XINP).GE.1.AND.INT(XINP).LE.3) THEN
      YINP=3.
    ELSE
      YINP=0.
    END IF
  ELSE IF (ABS(IMAP).EQ.1) THEN
    IF (IMAP.GT.0) THEN
      CALL MAPTRA (YINP,XINP,XOTP,YOTP)
    ELSE
      CALL MAPTRI (XINP,YINP,YOTP,XOTP)
    END IF
  ELSE IF (ABS(IMAP).EQ.2) THEN
    IF (IMAP.GT.0) THEN
      XOTP=XINP*COS(.017453292519943*YINP)
      YOTP=XINP*SIN(.017453292519943*YINP)
    ELSE
      XOTP=SQRT(XINP*XINP+YINP*YINP)
      YOTP=57.2957795130823*ATAN2(YINP,XINP)
    END IF
  ELSE
    XOTP=XINP
    YOTP=YINP
  END IF
  RETURN
END
```

When CPMPXY is called with IMAP = 0, it assumes it is being asked to return information about its mapping capabilities. XINP is assumed to have been given the value REAL(I), where I is the index of a mapping about which information is desired. CPMPXY sets YINP to indicate whether the mapping selected by I is implemented or not and whether its inverse is implemented or not. In the case of the default version of CPMPXY, mappings 1 through 3 are completely implemented (both forward and reverse), so a “3.” is returned as the value of YINP; other mappings are not implemented at all, so a “0.” is returned as the value of YINP.

When CPMPXY is called with IMAP = 1, the incoming X and Y coordinates are assumed to represent longitude and latitude, respectively; the EZMAP routine MAPTRA is called to find the X and Y coordinates of the projection of the specified point on the globe, and those coordinates are returned as the outgoing X and Y coordinates. When IMAP = -1, the incoming X and Y coordinates are assumed to be the X and Y coordinates of a projected point; the EZMAP routine MAPTRI is called to find the longitude and latitude of the original point on the globe, and those values are returned as the outgoing X and Y coordinates.

When IMAP = 2, the incoming X and Y coordinates are assumed to represent rho and theta (in degrees) in polar coordinates; from these are computed the output X and Y coordinates. When IMAP = -2, the incoming X and Y coordinates are used to compute rho and theta and those values are returned.

If IMAP is anything else, the input X and Y coordinates are simply returned as the output X and Y coordinates.

A user version of CPMPXY can be made to do any desired mapping. It should also be made, when IMAP = 0, to return correct information about its own capabilities.

3.20.2 Arguments

IMAP (INTEGER, input) is zero if the object of the call is to ask CPMPXY about its mapping capabilities, greater than zero if the object of the call is to do a forward mapping, and less than zero if the object of the call is to do an inverse mapping. When IMAP is nonzero, its absolute value matches the current value of the parameter 'MAP' and identifies the mapping to be used.

XINP (REAL, input) is used in one of three ways:

- When IMAP is zero, the value INT(XINP) is the index of a mapping about which CPMPXY is being asked to supply information.
- When IMAP is greater than zero, XINP is the X coordinate of a point on the contour plot. If 'XC1' = 'XCM' (the default situation), then XINP will lie in the range from 1 to M, where M is the first dimension of the array being contoured (equal to the value of the parameter 'ZDM'); in this case, the X coordinate will have the same range as the first index of the data array. If the user sets 'XC1' unequal to 'XCM', then XINP will lie in the range from 'XC1' (corresponding to an index value of 1) to 'XCM' (corresponding to an index value of M).
- When IMAP is less than zero, XINP is the X coordinate of a point on the contour plot, in a coordinate system consistent with the current window, as specified by arguments 5 through 8 of the last call to the SPPS routine SET or by the equivalent call to GKS.

YINP (REAL, input/output) is used in one of three ways:

- When IMAP is zero, CPMPXY is expected to return one of the following values of YINP: YINP = 0. indicates that neither the forward nor the inverse transformation is defined. YINP = 1. indicates that the forward transformation is defined, but the inverse is not. YINP = 2. indicates that the forward transformation is not defined, but the inverse is. YINP = 3. indicates that both the forward and the inverse transformations are defined.
- When IMAP is greater than zero, YINP is the Y coordinate of a point on the contour plot. If 'YC1' = 'YCN' (the default situation), then YINP will lie in the range from 1 to N, where N is the second dimension of the array being contoured (equal to the value of the parameter 'ZDN'); in this case, the Y coordinate will have the same range as the second index of the data array. If the user sets 'YC1' unequal to 'YCN', then YINP will lie in the range from 'YC1' (corresponding to an index value of 1) to 'YCN' (corresponding to an index value of N).
- When IMAP is less than zero, YINP is the Y coordinate of a point on the contour plot, in a coordinate system consistent with the current window, as specified by arguments 5 through 8 of the last call to the SPPS routine SET or by the equivalent call to GKS.

XOTP (REAL, output) and **YOTP** (REAL, output) are used in one of two ways:

- If IMAP is greater than zero, XOTP and YOTP are the X and Y coordinates of a point on the contour plot, in a coordinate system consistent with the current window, as specified by arguments 5 through 8 of the last call to the SPPS routine SET or by the equivalent call to GKS.
- When IMAP is less than zero, XOTP and YOTP are the X and Y coordinates of a point on the contour plot. If 'XC1' = 'XCM' (the default situation), then XOTP will lie in the range from 1 to M, where M is the first dimension of the array being contoured (equal to the value of the parameter 'ZDM'); in this case, the X coordinate will have the same range as the first index of the data array. If the user sets 'XC1' unequal to 'XCM', then XOTP will lie in the range from 'XC1' (corresponding to an index value of 1) to 'XCM' (corresponding to an index value of M). Similarly, if 'YC1' = 'YCN' (the default situation), then YOTP will lie in the range from 1 to N, where N is the second dimension of the array being contoured (equal to the value of the parameter 'ZDN'); in this case, the Y coordinate will have the same range as the second index of the data array. If the user sets 'YC1' unequal to 'YCN', then YOTP will lie in the range from 'YC1' (corresponding to an index value of 1) to 'YCN' (corresponding to an index value of N).

In any case, if the point (XINP,YINP) cannot be mapped for any reason, some recognizable impossible value should be returned for both of XOTP and YOTP and the internal parameter 'ORV' should be given that value, thereby allowing CONPACK routines that call CPMPXY to determine whether or not a point being projected is visible or not. The value used for this purpose by the EZMAP routines MAPTRA and MAPTRI is 1.E12.

3.21 CPMVIW (IWKO,IWKN,LWKN)

This routine is called to move what CONPACK has in the current integer workspace array to a new array.

3.21.1 Usage

When, in the execution of a CONPACK routine, the amount of space left in the integer workspace array is found to be insufficient, if the internal parameter 'WSO' has the value 3, the error-handling routine SETER is called with an appropriate error message. If, in addition, the user has turned recovery mode on, execution continues and, eventually, control is returned to the user. At that point, the user should detect the fact that an error has occurred. If he/she chooses to try to recover from the error, CPMVIW may be of use: it may be called to move everything from the current integer workspace array to a new (and presumably bigger) one, after which it may be possible to resume execution.

3.21.2 Arguments

IWKO (an input array of type INTEGER, dimensioned as specified in a previous call to CPRECT, CPSPS1, CPSPS2, or CPMVIW) is the current (“old”) integer workspace array.

IWKN (an output array of type INTEGER, dimensioned LWKN) is the array which is to become the new integer workspace array.

LWKN (an input expression of type INTEGER) is the dimension of the array IWKN.

3.22 CPMVRW (RWKO,RWKN,LWKN)

This routine is called to move what CONPACK has in the current real workspace array to a new array.

3.22.1 Usage

When, in the execution of a CONPACK routine, the amount of space left in the real workspace array is found to be insufficient, if the internal parameter 'WSO' has the value 3, the error-handling routine SETER is called with an appropriate error message. If, in addition, the user has turned recovery mode on, execution continues and, eventually, control is returned to the user. At that point, the user should detect the fact that an error has occurred. If he/she chooses to try to recover from the error, CPMVRW may be of use: it may be called to move everything from the current real workspace array to a new (and presumably bigger) one, after which it may be possible to resume execution.

3.22.2 Arguments

RWKO (an input array of type REAL, dimensioned as specified in a previous call to CPRECT, CPSPS1, CPSPS2, or CPMVRW) is the current (“old”) real workspace array.

RWKN (an output array of type REAL, dimensioned LWKN) is the array which is to become the new real workspace array.

LWKN (an input expression of type INTEGER) is the dimension of the array RWKN.

3.23 CPPKCL (ZDAT,RWRK,IWRK)

This routine is called to pick a set of contour levels.

3.23.1 Usage

Normally, CPPKCL is called by CONPACK when the contour levels are needed; it may be called by the user (after the initialization call to CPRECT, CPSPS1, or CPSPS2) when it is desired to modify the resulting parameter arrays specifying contour levels and associated quantities in some way.

If the contour-level-selection parameter 'CLS' is zero, CPPKCL does nothing. If the constant-field-flag 'CFF' is non-zero, indicating that, on the last call to CPRECT, CPSPS1, or CPSPS2, the data were found to be essentially constant, CPPKCL does nothing.

If neither of the conditions mentioned in the last paragraph arises, contour levels are picked. For a complete description of the various possible ways in which this might be done, see the description of the parameter 'CLS'. In any case, 'NCL' is set to the number of contour levels picked and elements 1 through 'NCL' of the parameter array 'CLV' are set to the levels themselves. The Ith elements of the associated parameter arrays are set as follows:

```
'CLU' = 1 OR 3              CONTOUR LEVEL USE FLAG
'AIA' = I+1                 AREA IDENTIFIER ABOVE THE LEVEL
'AIB' = I                   AREA IDENTIFIER BELOW THE LEVEL
'LLT' = ' '                 CONTOUR LINE LABEL TEXT
'CLD' = '$$$$$$$$$$$$$$$$'  CONTOUR LINE DASH PATTERN
'CLC' = -1                  CONTOUR LINE COLOR
'LLC' = -1                  CONTOUR LINE LABEL COLOR
'CLL' = -1.                 CONTOUR LINE LINE WIDTH
```

Thus, after CPPKCL is called, the situation for each contour level is this: Contour lines at the level are to be generated by calls to CPCLDM or CPLBDR; the lines will be labelled if 'CLU' = 3, but not if 'CLU' = 1. If CPCLAM is called, the lines will be added to the area map; the identifier used for areas “above” the lines will be the contour level index plus one and the identifier used for areas “below” the lines will be the contour level index. The text of the label associated with the level is, as yet, unspecified (see the description of the subroutine CPPKLB, below). The dash pattern for the level is solid, and neither the color of the line, the color of the labels on it, nor its width are to be set. By resetting elements in these parameter arrays, the user can change the situation in various ways.

3.23.2 Arguments

All three arguments are arrays used in a previous call to one of the routines CPRECT, CPSPS1, CPSPS2, CPMVRW, or CPMVIW, the contents of which must not have been changed by the user since that call.

ZDAT (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, or CPSPS2, input) is the data array.

RWRK (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVRW, input/output) is the current real workspace array.

IWRK (INTEGER array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVIW, input/output) is the current integer workspace array.

3.24 CPPKLB (ZDAT,RWRK,IWRK)

This routine is called to pick a set of labels for labelled contour levels.

3.24.1 Usage

Normally, CPPKLB is called by CONPACK when labels for the contour levels are needed; it may be called by the user (after the initialization call to CPRECT, CPSPS1, or CPSPS2) when it is desired to modify the resulting parameter arrays specifying the labels.

If the constant-field-flag 'CFF' is non-zero, indicating that, on the last call to CPRECT, CPSPS1, or CPSPS2, the data were found to be essentially constant, CPPKLB does nothing; otherwise, it examines the first 'NCL' elements of the parameter array 'CLV', which defines the contour levels, and the associated parameter arrays, looking for levels which are to be labelled ('CLU' = 2 or 3) for which no label is specified ('LLT' = ' ', a single blank). If any such levels are found, labels are generated for them.

The scale factor 'SFU' may be set as a byproduct of choosing the labels. See the description of the parameter 'SFS'.

After calling CPPKLB, a user program may examine the generated labels and change them in various ways.

3.24.2 Arguments

All three arguments are arrays used in a previous call to one of the routines CPRECT, CPSPS1, CPSPS2, CPMVRW, or CPMVIW, the contents of which must not have been changed by the user since that call.

ZDAT (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, or CPSPS2, input) is the data array.

RWRK (REAL array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVRW, input/output) is the current real workspace array.

IWRK (INTEGER array, dimensioned as specified in the last call to CPRECT, CPSPS1, CPSPS2, or CPMVIW, input/output) is the current integer workspace array.

3.25 CPRECT (ZDAT,KZDT,MZDT,NZDT, ...)

(The remaining arguments are **RWRK**, **LRWK**, **IWRK**, and **LIWK**.)

Initializes the contouring of a rectangular array of data.

3.25.1 Usage

The routine CPRECT is called to initialize the process of drawing a contour plot from a rectangular array of data. The arguments define the data array, a real workspace array, and an integer workspace array. The dimensions of all the arrays are transferred to variables in COMMON, so that, in calls to other CONPACK routines, those dimensions may be omitted. The internal pointers which are used to manage workspace use are initialized. It is decided what the ranges of X and Y coordinates used to draw contour lines and position labels ought to be. If CONPACK is to call SET, appropriate arguments are determined and SET is called; otherwise, GETSET is called to retrieve the arguments from the user's call to SET. The list of label positions (if any) left over from previous calls to the package is discarded. If contour levels are to be chosen by the package, 'NCL' is zeroed so that the levels will be chosen when required. The minumum and maximum values in the data array are located and it is decided whether the data are essentially constant. Numeric-label parameters which depend on the range of values in the data array are initialized. A scale factor may be chosen.

3.25.2 Arguments

ZDAT (REAL array, dimensioned KZDT x n, where “n” is greater than or equal to NZDT, input) is the array of data to be contoured.

KZDT (INTEGER, input) is the first dimension of the array ZDAT.

MZDT (INTEGER, input) is the first dimension of the array of data in ZDAT. MZDT must be less than or equal to KZDT.

NZDT (INTEGER, input) is the second dimension of the array of data in ZDAT. NZDT must be less than or equal to the declared second dimension of the array ZDAT.

RWRK (REAL array, dimensioned LRWK, input/output) is the real work array.

LRWK (INTEGER, input) is the length of RWRK.

IWRK (INTEGER array, dimensioned LIWK, input/output) is the integer work array.

LIWK (INTEGER, input) is the length of IWRK.

3.26 CPRSET

Resets all parameters to their initial default values.

3.26.1 Usage

Use “CALL CPRSET” to reset all parameters to their default values.

3.26.2 Arguments

None.

3.27 CPSCAE (ICRA,ICA1,ICAM,ICAN, ...)

(The remaining arguments are **XCPF**, **YCPF**, **XCQF**, **YCQF**, **IND1**, **IND2**, **ICAF**, and **IAID**.)

Updates a particular element of a user’s cell array.

3.27.1 Usage

CPSCAE is not called by the user. When a user calls the routine CPCICA and the internal parameter 'CAF' is negative, CPSCAE is called by CPCICA exactly once for each cell in the cell array. The default version of CPSCAE just sets the cell’s color index equal to the area identifier associated with the cell, if that area identifier is greater than or equal to zero; otherwise, it does nothing. A user-supplied version of CPSCAE may do something more complicated; for example, it may take into account both the value of the area identifier associated with the cell and the current value of the color index for the cell in determining the new value of the color index for the cell.

3.27.2 Arguments

The first eight arguments of CPSCAE describe the user's cell array and are taken directly from the user's call to CPCICA:

ICRA (INTEGER array, dimensioned ICA1 by "n", where "n" is greater than or equal to the value of the argument ICAN, input/output) is the user's cell array. Generally speaking, a user version of CPSCAE should only reset the value of the single element ICRA(IND1,-IND2).

ICA1 (INTEGER, input) is the first dimension of the FORTRAN array ICRA, which contains the user's cell array.

ICAM (INTEGER, input) is the first dimension of the user's cell array.

ICAN (INTEGER, input) is the second dimension of the user's cell array.

XCPF and **YCPF** (REAL, input) are the coordinates, in the fractional coordinate system, of a point P. P is the point at that corner of the rectangular area into which the cell array maps that corresponds to the cell (1,1).

XCQF and **YCQF** (REAL, input) are the coordinates, in the fractional coordinate system, of a point Q. Q is the point at that corner of the rectangular area into which the cell array maps that corresponds to the cell (ICAM,ICAN).

IND1 and **IND2** (INTEGER, input) are the indices of an element in the cell array that is to be reset.

ICAF (INTEGER, input) is the value of the internal parameter 'CAF', which will be a negative integer (because, when 'CAF' is greater than or equal to zero, the routine CPSCAE is not called). Different values of 'CAF' may be used in a user version of CPSCAE to select different schemes for picking color indices.

IAID (INTEGER, input) is the value of the area identifier associated with the midpoint of the cell (IND1,IND2). The four possibilities are described above, in the description of the routine CPCICA.

3.28 CPSETC (PNAM,CVAL)

This routine is called to set the value of a parameter of type CHARACTER.

3.28.1 Usage

Use the statement

```
CALL CPSETC (PNAM,CVAL)
```

to give the parameter whose name is PNAM the character value CVAL. If that parameter is an array, the element specified by the current value of 'PAI' will be the one changed.

3.28.2 Arguments

PNAM (CHARACTER, input) is the name of a parameter to be given a character value. Only the first three characters of PNAM are examined. It is recommended that the rest of the character string be used to improve the readability of the code. For example, instead of 'ILT', use 'ILT - INFORMATIONAL LABEL TEXT'.

CVAL (CHARACTER, input) is a character constant or variable, the value of which is to be given to the parameter specified by PNAM.

3.29 CPSETI (PNAM,IVAL)

This routine is called to give a new integer value to an internal parameter.

3.29.1 Usage

Use the statement

```
CALL CPSETI (PNAM,IVAL)
```

to give the parameter whose name is PNAM the integer value IVAL. If that parameter is an array, the element specified by the current value of 'PAI' will be the one changed.

3.29.2 Arguments

PNAM (CHARACTER, input) is the name of a parameter to be given an integer value. Only the first three characters of PNAM are examined. It is recommended that the rest of the character string be used to improve the readability of the code. For example, instead of 'ILA', use 'ILA - INFORMATIONAL LABEL ANGLE'.

IVAL (INTEGER, input) is an expression, the value of which is to be given to the parameter specified by PNAM. If the internal parameter is of type INTEGER, it is given the value IVAL; if the internal parameter is of type REAL, it is given the value REAL(IVAL).

3.30 CPSETR (PNAM,RVAL)

This routine is called to give a new real value to an internal parameter.

3.30.1 Usage

Use the statement

```
CALL CPSETR (PNAM,RVAL)
```

to give the parameter whose name is PNAM the real value RVAL. If that parameter is an array, the element specified by the current value of 'PAI' will be the one changed.

3.30.2 Arguments

PNAM (CHARACTER, input) is the name of a parameter to be given a real value. Only the first three characters of PNAM are examined. It is recommended that the rest of the character string be used to improve the readability of the code. For example, instead of 'CIS', use 'CIS - CONTOUR INTERVAL SPECIFIER'.

RVAL (REAL, input) is an expression, the value of which is to be given to the parameter specified by PNAM. If the internal parameter is of type INTEGER, it is given the value INT(RVAL); if the internal parameter is of type REAL, it is given the value RVAL.

3.31 CPSPS1 (ZSPS,KSPS,MSPS,NSPS, ...)

(The remaining arguments are **RWRK**, **LRWK**, **IWRK**, **LIWK**, **ZDAT**, and **LZDT**.)

This routine is called to interpolate from an array of data on a “sparse” rectangular grid which is regularly spaced in X and Y to an array of data on a “dense” rectangular grid and to initialize contouring from the array on the dense grid. (By a “sparse” grid is meant one whose dimensions are smaller than one would like, so that contour lines constructed directly on it are composed of long straight segments.) CPSPS1 may be viewed as a data smoothing routine.

3.31.1 Usage

The routine CPSPS1 is called to initialize the process of drawing a contour plot from a sparse array of data. The arguments define the sparse array, a real workspace array, an integer workspace array, and an array in which the dense array of data is to be formed and returned to the user. The dimensions of all the arrays except the first are transferred to variables in COMMON, so that, in calls to other CONPACK routines, these dimensions may be omitted. The internal pointers which are used to manage workspace use are initialized. It is decided what the ranges of X and Y coordinates used to draw contour lines ought to be. If CONPACK is to call SET, appropriate arguments are determined and SET is called; otherwise, GETSET is called to retrieve the arguments from the user’s call to SET. Label-position information left over from previous calls to the package is discarded. If contour levels are to be chosen by the package, 'NCL' is zeroed so that the levels will be chosen when required. The minumum and maximum values in the dense data array are located and it is decided whether the data are essentially constant. Numeric-label parameters which depend on the range of values in the dense array are initialized. A scale factor may be chosen.

The routines BSURF1 and BSURF2, from the package FITPACK, by Alan K. Cline, are used to fit bicubic splines under tension to the “sparse” array of data and to compute the dense grid of data which is returned to the user. The tension on the spline surfaces is specified by the parameter 'T3D'. By default, CPSPS1 selects the dimensions of the dense array of data; if desired, these dimensions may be specified by the user (by setting the parameter 'ZDS' non-zero and the parameters 'ZD1', 'ZDM', and 'ZDN' to the desired values). In either case, 'ZD1', 'ZDM', and 'ZDN' are set and should not be reset by the user until the contour plot is complete and a different contour plot is to be drawn.

Because the routines BSURF1 and BSURF2 do not have a built-in “special value” feature, if the special-value parameter 'SPV' is non-zero and the sparse array contains occurrences of that value, special action must be taken. The indices of the special values in the sparse array are saved in a part of the integer workspace array; the special values are then replaced by values interpolated from adjacent grid points and the resulting array is used to obtain the dense array. Then, the special values in the sparse array are restored and the corresponding elements of the dense array are also given the special value.

3.31.2 Arguments

ZSPS (REAL array, dimensioned KSPS x n, where “n” is greater than or equal to NSPS, input) is the “sparse” array of data, from which the “dense” array is to be generated.

KSPS (INTEGER, input) is the first dimension of the array ZSPS.

MSPS (INTEGER, input) is the first dimension of the “sparse” array of data in ZSPS. MSPS must be less than or equal to KSPS.

NSPS (INTEGER, input) is the second dimension of the “sparse” array of data in ZSPS. NSPS must be less than or equal to the declared second dimension of the array ZSPS.

RWRK (REAL array, dimensioned LRWK, input/output) is the real work array.

LRWK (INTEGER, input) is the length of RWRK.

IWRK (INTEGER array, dimensioned LIWK, input/output) is the integer work array.

LIWK (INTEGER, input) is the length of IWRK.

ZDAT (REAL array, dimensioned LZDT, output) is the array in which the interpolated “dense” array of data is to be returned. The dimensions of the interpolated array may be supplied by the user or determined by CONPACK, depending on the value of the parameter 'ZDS'. Note that, if the size of the dense array is not a product of the size of the sparse array and some perfect square, the aspect ratio of the dense grid may be slightly different from that of the sparse grid.

LZDT (INTEGER, input) is the length of ZDAT.

3.32 CPSPS2 (XSPS,YSPS,ZSPS,KSPS,MSPS,NSPS, ...)

(The remaining arguments are **RWRK**, **LRWK**, **IWRK**, **LIWK**, **ZDAT**, **LZDT**.)

This routine is called to interpolate from an array of data on a “sparse” rectangular grid which is irregularly spaced in X and Y to an array of data on a “dense” rectangular grid and to initialize contouring from the array on the dense grid. (By a “sparse” grid is meant one whose dimensions are smaller than one would like, so that contour lines constructed directly on it are composed of long straight segments.) CPSPS2 may be viewed as a data smoothing routine.

3.32.1 Usage

The routine CPSPS2 is called to initialize the process of drawing a contour plot from a sparse array of data. The arguments define the X and Y coordinates of the irregular grid, the sparse array defined on that grid, a real workspace array, an integer workspace array, and an array in which the dense array of data is to be formed and returned to the user. The dimensions of all the arrays except the first three are transferred to variables in COMMON, so that, in calls to other CONPACK routines, these dimensions may be omitted. The internal pointers that are used to manage workspace use are initialized. It is decided what the ranges of X and Y coordinates used to draw contour lines ought to be. If CONPACK is to call SET, appropriate arguments are determined and SET is called; otherwise, GETSET is called to retrieve the arguments from the user’s call to SET. Label-position information left over from previous calls to the package is discarded. If contour levels are to be chosen by the package, 'NCL' is zeroed so that the levels will be chosen when required. The minumum and maximum values in the dense data array are located and it is decided whether the data are essentially constant. Numeric-label parameters which depend on the range of values in the dense array are initialized. A scale factor may be chosen.

The routines SURF1 and SURF2, from the package FITPACK, by Alan K. Cline, are used to fit bicubic splines under tension to the “sparse” array of data and to compute the dense grid of data which is returned to the user. The tension on the spline surfaces is specified by the parameter 'T3D'. By default, CPSPS2 selects the dimensions of the dense array of data; if desired, these dimensions may be specified by the user (by setting the parameter 'ZDS' non-zero and the parameters 'ZD1', 'ZDM', and 'ZDN' to the desired values). In either case, 'ZD1', 'ZDM', and 'ZDN' are set and should not be reset by the user until the contour plot is complete and a different contour plot is to be drawn.

Because the routines SURF1 and SURF2 do not have a built-in “special value” feature, if the special-value parameter 'SPV' is non-zero and the sparse array contains occurrences of that value, special action must be taken. The indices of the special values in the sparse array are saved in a part of the integer workspace array; the special values are then replaced by values interpolated from adjacent grid points and the resulting array is used to obtain the dense array. Then, the special values in the sparse array are restored and the corresponding elements of the dense array are also given the special value.

3.32.2 Arguments

XSPS (REAL array, dimensioned MSPS) is the array of X coordinates of the irregular rectangular grid. These must be in strictly increasing numerical order.

YSPS (REAL array, dimensioned NSPS) is the array of Y coordinates of the irregular rectangular grid. These must be in strictly increasing numerical order.

ZSPS (REAL array, dimensioned KSPS x n, where "n" is greater than or equal to NSPS, input) is the "sparse" array of data, from which the "dense" array is to be generated.

KSPS (INTEGER, input) is the first dimension of the array ZSPS.

MSPS (INTEGER, input) is the first dimension of the "sparse" array of data in ZSPS. MSPS must be less than or equal to KSPS.

NSPS (INTEGER, input) is the second dimension of the "sparse" array of data in ZSPS. NSPS must be less than or equal to the declared second dimension of the array ZSPS.

RWRK (REAL array, dimensioned LRWK, input/output) is the real work array.

LRWK (INTEGER, input) is the length of RWRK.

IWRK (INTEGER array, dimensioned LIWK, input/output) is the integer work array.

LIWK (INTEGER, input) is the length of IWRK.

ZDAT (REAL array, dimensioned LZDT, output) is the array in which the interpolated "dense" array of data is to be returned. The dimensions of the interpolated array may be supplied by the user or determined by CONPACK, depending on the value of the parameter 'ZDS'. Note that, if CONPACK determines the dimensions of the "dense" array, it will attempt to use an aspect ratio which is close to that implied by the value of the ratio

```
(XSPS(MSPS)-XSPS(1))/(YSPS(NSPS)-YSPS(1))
```

LZDT (INTEGER, input) is the length of ZDAT.

4.0 PARAMETERS

The behavior of CONPACK is controlled by a large number of parameters in internal common blocks to which all of its routines have access. These parameters are described in detail in this section.

4.1 Parameter Names

Each of the parameters of CONPACK is identified by a three-character mnemonic name. For example, 'NCL' is the name of the parameter specifying the number of contour levels and 'SPV' is the name of a parameter which, if non-zero, specifies a “special value”, used to indicate missing data in the arrays of data being contoured.

4.2 Parameter Types

Each parameter is intrinsically of type CHARACTER, INTEGER, or REAL, but the type of the parameter is not implied by the form of its name; one must read the description of the parameter to determine its type.

4.3 Parameter Defaults

Each parameter has a default value which is intended to yield a reasonable, useful, default behavior. To modify the behavior of CONPACK routines, one changes the values of the appropriate parameters at the appropriate time.

4.4 Parameter Access

The value of a parameter may be set using one of the three routines CPSETC, CPSETI, and CPSETR. Each of these routines has as its first argument a character string beginning with the three-character name of the parameter and as its second argument an expression of the type implied by the last character of the routine name (“C” for “CHARACTER”, “I” for “INTEGER”, or “R” for “REAL”). Similarly, the current value of a parameter may be retrieved using one of the three routines CPGETC, CPGETI, and CPGETR, each of which has as its first argument the name of the parameter and as its second argument a variable of the type implied by the last character of the routine name.

In general, once a parameter is given a particular value, it retains that value until it is given a new value; with only a couple of exceptions, there is no automatic resetting of parameters at any time.

4.5 Parameter-Name Arguments

Only the first three characters of the parameter-name argument in calls to the parameter access routines are examined. It is strongly recommended that additional characters be added in order to make the code more readable. For example, a call to set the number of contour levels might read

```
CALL CPSETI ('NCL - NUMBER OF CONTOUR LEVELS',13)
```

The additional characters make it much easier to remember what the call is intended to do.

4.6 Automatic Type Conversion

Normally, one would use CPSETI to set parameters of type INTEGER and CPSETR to set parameters of type REAL. However, since automatic conversion is done as necessary, this need not be the case. One could, for example, use either of the two statements

```
CALL CPSETI ('ILA - INFORMATIONAL LABEL ANGLE',30)
CALL CPSETR ('ILA - INFORMATIONAL LABEL ANGLE',30.)
```

to set the informational-label angle (which is intrinsically REAL) to 30 degrees. The first has the virtue of being one character shorter. Similarly, one could use either of the two statements

```
CALL CPSETI ('NCL - NUMBER OF CONTOUR LEVELS',18)
CALL CPSETR ('NCL - NUMBER OF CONTOUR LEVELS',18.)
```

to set the number of contour levels (which is intrinsically INTEGER) to 18. (In this case, of course, the first of the two statements seems to make better sense.) If a REAL parameter is to be given a non-integral value (like 3.14159), then CPSETR must be used, of course.

Certain parameters, of type CHARACTER, representing dash patterns, may be set using either a call to CPSETC or CPSETI; in the latter case, the low-order sixteen bits of the integer are interpreted and used to construct a character dash pattern (by mapping 0’s into apostrophes and 1’s into dollar signs).

4.7 Automatic Restriction of Parameter Values

Some parameters may take on any possible value of the proper type. For example, 'CFA', which specifies the angle at which the constant-field label is to be written, may be given any real value. Other parameters may only be given values in a restricted range. For example, 'CFB', which specifies whether the constant-field label is to be boxed, and, if so, in what manner, is restricted to the integer values 0, 1, 2, and 3. The parameter-setting routines attempt to enforce such restrictions. Attempting to set 'CFB' to a non-integral value will result in truncation of the fractional part; attempting to give it a value less than 0 will result in giving it the value 0 and attempting to give it a value greater than 3 will result in giving it the value 3. Other parameters are treated similarly.

4.8 Parameter Arrays

Some of the parameters are actually arrays. For example, the parameter 'CLV', which specifies contour levels, is actually a 256-element array. In order to access the Ith element of such a parameter array, one must somehow specify the value of the index I. This is done by making one parameter, named 'PAI', serve as a “parameter array index”. Thus, to set the tenth contour level to 196.3, one would use the following code:

```
CALL CPSETI ('PAI - PARAMETER ARRAY INDEX',10)
CALL CPSETR ('CLV - CONTOUR LEVEL VALUE',196.3)
```

In a few cases, negative values of PAI must be used. A negative value of PAI is not used directly as an index value; it merely serves as a signal to the parameter-access routines that a special element of the parameter array is being accessed. These cases will be pointed out as they are encountered.

4.9 Parameter Descriptions

Parameter descriptions follow, in alphabetical order. Each description begins with a line giving the three-character mnemonic name of the parameter, the phrase for which the mnemonic stands, the intrinsic type of the parameter, and an indication of whether or not it is an array.

4.9.1 'AIA' - Area Identifier Above - Integer Array

Each element of the parameter array 'AIA' is an area identifier for the area above the contour level specified by the corresponding element of the parameter array 'CLV'. The corresponding element of the parameter array 'AIB' is an area identifier for the area below that level. If, for a particular contour level, both 'AIA' and 'AIB' are zero, that level is ignored by the routine CPCLAM; otherwise, contour lines at that level are added to the area map and the given values of 'AIA' and 'AIB' are used as right and left area identifiers.

There are three special elements in the parameter array 'AIA', corresponding to 'PAI' values of -1, -2, and -3; the first specifies an area identifier for the area outside the edge of the grid, the second an area identifier for any area filled with special values, and the third an area identifier for any area in which the mapping routine CPMPXY returns the “out of range” value 'ORV'.

If contour levels are chosen automatically, rather than being supplied by the user, the value supplied for the Ith element of 'AIA' is I+1 and the value supplied for the Ith element of 'AIB' is I.

The default value of element “-1” of 'AIA' is 0 and the default values of elements “-2” and “-3” are -1’s.

4.9.2 'AIB' - Area Identifier Below - Integer Array

See the description of 'AIA', above.

4.9.3 'CAF' - Cell Array Flag - Integer

The parameter 'CAF' determines the way in which the routine CPCICA (which is called to store color indices in a user cell array), gets from an area identifier associated with a particular cell to a color index for that cell. Let IAID be the area identifier that is associated with a given cell. Then:

- If 'CAF' is negative, the routine CPSCAE is called. The default version of CPSCAE behaves as follows: if the value of IAID is greater than or equal to zero, it itself is stored in the cell array as a color index; otherwise, the cell array is unchanged. A user version of CPSCAE may be supplied to do something more complicated; since the value of 'CAF' is one of the arguments of CPSCAE, its value may be used to select one of a number of color-setting schemes built into the routine.

- If 'CAF' is positive, the value "'CAF'+IAID" is computed. If that value is positive, it is stored in the cell array as a color index; otherwise, the cell array is unchanged. In this case, 'CAF' acts as an offset, mapping the area identifiers to a unique portion of the color table.

The default value of 'CAF' is 0.

4.9.4 'CFA' - Constant-Field Label Angle - Real

The parameter 'CFA' specifies the angle (in degrees counterclockwise from a vector pointing to the right) at which a constant-field label is to be written.

The default value of 'CFA' is 0.

4.9.5 'CFB' - Constant-Field Label Box Flag - Integer

If 'CFB' is zero, the constant-field label will not be boxed at all. The value 1 implies that the perimeter of the box is to be drawn (in the same color as the label) after the label is drawn. The value 2 implies that the box is to be filled (in the color specified by 'LBC') before the label is drawn. The value 3 implies both of the above.

The default value of 'CFB' is 0.

4.9.6 'CFC' - Constant-Field Label Color Index - Integer

If 'CFC' is less than zero, the constant-field label and the box, if any, around it, will be drawn in the color specified by the current text color index; if 'CFC' is greater than or equal to zero, then it specifies the desired color index for the label and the box. If a box is drawn around the label, it is made the same color as the label itself.

The default value of 'CFC' is -1.

4.9.7 'CFF' - Constant-Field-Found Flag - Integer

The parameter 'CFF' may not be set by the user; its retrieved value will be non-zero if and only if CPRECT, CPSPS1, or CPSPS2 detected a constant field.

The default value of 'CFF' (prior to any call to CPRECT, CPSPS1, or CPSPS2) is zero.

4.9.8 'CFL' - Constant-Field Label Line Width - Real

If 'CFL' is less than or equal to zero, line width will not be set before drawing a box around the constant-field label. If 'CFL' is greater than zero, it specifies the desired width, as a multiple of the "normal" line width.

The default value of 'CFL' is 0.

4.9.9 'CFP' - Constant-Field Label Positioning Flag - Integer

The parameter 'CFP' says how the constant-field label is to be positioned. There are nine possible values, each of which specifies a point of the label box which is to lie on the point defined by 'CFX' and 'CFY': the value -4 implies the lower left-hand corner of the label box, -3 implies the center of the bottom of the box, -2 the lower right-hand corner of the box, -1 the center of the left edge of the box, 0 the center of the box, +1 the center of the right edge of the box, +2 the upper left-hand corner of the box, +3 the center of the top edge of the box, and +4 the upper right-hand corner of the box. Left, right, bottom, and top are defined from the viewpoint of someone viewing the label right-side up.

The default value of 'CFP' is 0, so the constant-field label will be centered on the point whose coordinates are 'CFX' and 'CFY'.

4.9.10 'CFS' - Constant-Field Label Size - Real

The parameter 'CFS' specifies the nominal size (width) of a character in the constant-field label, as a fraction of the width of the viewport. This nominal size is multiplied by 'CWM'.

The default value of 'CFS' is .012.

4.9.11 'CFT' - Constant-Field Label Text String - Character

The parameter 'CFT' specifies the text of the constant-field label, which is written when a constant data field is detected; it is a character string of at most 64 characters. The embedded string 'ZDV' will be replaced by the numeric value of the field.

If 'CFT' is given the value ' ' (a single blank), the constant-field label will not be written.

The default value of 'CFT' is 'CONSTANT FIELD - VALUE IS ZDV'.

4.9.12 'CFW' - Constant-Field Label White Space Width - Real

The parameter 'CFW' specifies the nominal width of white space to be left around the constant-field label, as a fraction of the width of the viewport. This nominal width is multiplied by 'CWM'.

The default value of 'CFW' is .005.

4.9.13 'CFX' - Constant-Field Label X Coordinate - Real

The parameter 'CFX' specifies the X coordinate of the basepoint of the constant-field label. The given value is mapped linearly onto the viewport; 0 refers to the left edge of the viewport and 1 to the right edge of the viewport. Values less than 0 or greater than 1 may be used.

The default value of 'CFX' is .5, so the constant-field label is centered horizontally in the viewport.

4.9.14 'CFY' - Constant-Field Label Y Coordinate - Real

The parameter 'CFY' specifies the Y coordinate of the basepoint of the constant-field label. The given value is mapped linearly onto the viewport; 0 refers to the bottom edge of the viewport and 1 to the top edge of the viewport. Values less than 0 or greater than 1 may be used.

The default value of 'CFY' is .5, so the constant-field label is centered vertically in the viewport.

4.9.15 'CIS' - Contour Interval Specifier - Real

See the description of 'CLS', below. When 'CLS' is greater than zero, 'CIS' is used. A value of 'CIS' less than or equal to zero indicates that CONPACK is to choose the contour interval (see the descriptions of the parameter arrays 'CIT' and 'LIT'). A value of 'CIS' which is greater than zero is the actual contour interval to be used; in this case, 'LIS' may be given a non-zero value “n” to specify that every “nth” contour level should be labelled. See also the descriptions of 'CMN' and 'CMX', below.

The default value of 'CIS' is 0.

4.9.16 'CIT' - Contour Interval Table - Real Array

Each non-zero element of the ten-element parameter array 'CIT' is a real number greater than or equal to 1 and less than 10; when CONPACK picks the contour interval, it chooses a number of the form “c x 10^k”, where “c” is one of the elements of 'CIT' and “k” is an integer. The non-zero elements of 'CIT' must be sorted in ascending order and appear at the beginning of the array.

The corresponding elements of the parameter array 'LIT' specify which contours are to be labelled. A contour at “nc x 10^k” is labelled if “n” is a multiple of “l”, where “l” is the element of 'LIT' corresponding to the element “c” of 'CIT'. For example, if the first element of 'CIT' is “1.” and the first element of 'LIT' is “5”, then CONPACK is allowed to use contour levels 1., 2., 3., 4., etc., with labels at 5., 10., 15., etc. (5 x 1., 5 x 2., 5 x 3., etc.).

The default contents of 'CIT' and 'LIT' are as follows:

'PAI'	'CIT'	'LIT'	LEVELS ALLOWED	LABEL INTERVAL
1	1.0	5	1.0 x 10 ** K	EVERY 5TH ONE
2	2.0	5	2.0 x 10 ** K	EVERY 5TH ONE
3	2.5	4	2.5 x 10 ** K	EVERY 4TH ONE
4	4.0	5	4.0 x 10 ** K	EVERY 5TH ONE
5	5.0	5	5.0 x 10 ** K	EVERY 5TH ONE
6	0.0	0	none	none
7	0.0	0	none	none
8	0.0	0	none	none
9	0.0	0	none	none
10	0.0	0	none	none

4.9.17 'CIU' - Contour Interval Used - Real

The parameter 'CIU' is normally intended for retrieval only. When the selection of the contour interval is left up to CONPACK, 'CIU' is given the value chosen. When contour levels are completely set by the user, the value of 'CIU' may need to be set as well, for two reasons: 1) to make the desired value appear in an informational label (in place of the embedded string 'CIU') and 2) so that it may be used by the penalty scheme for positioning labels. The setting of 'CIU' must be done after setting the contour levels (because, as a side effect of the setting of element 1 of 'CLV', 'CIU' is zeroed). If the user supplies contour levels, but supplies no value of 'CIU', and the penalty scheme is used to position labels, the required contour interval is estimated; in certain situations, this can lead to problems (if, for example, the same contour level appears twice in 'CLV', once to force lines at that level to be drawn and once to force that level to be used as the boundary for a shaded area).

The default value of 'CIU' is 0.

4.9.18 'CLC' - Contour Line Color Index - Integer Array

Each element of the parameter array 'CLC', if greater than or equal to zero, is a color index for contour lines at the level specified by the associated element of 'CLV'. A value less than zero implies that the lines will be drawn in the color specified by the current polyline color index.

There are three special elements in the parameter array 'CLC', corresponding to 'PAI' values of -1, -2, and -3; the first specifies a color index for the edge of the grid, the second a color index for the edge of any area filled with special values, and the third a color index for the edge of any area in which the mapping routine CPMPXY returns the “out of range” value.

The default value of each element of 'CLC' is -1.

4.9.19 'CLD' - Contour Line Dash Pattern - Character Array

Each element of the parameter array 'CLD' is a dash pattern (as expected by the package DASHCHAR, when 'DPU' is greater than zero, or by the package DASHPACK, when 'DPU' is less than zero) to be used (when 'DPU' is non-zero) to draw contour lines at the level specified by the associated element of the contour level array 'CLV'. Elements of 'CLD' may be set using a call to CPSETI, with a sixteen-bit integer as the second argument, or using a call to CPSETC, with a character string of 128 or fewer characters as the second argument. In either case, the result will be a character string internally; a sixteen-bit integer will be converted to a sixteen-character string by mapping 0-bits into apostrophes and 1-bits into dollar signs. (When 'DPU' is negative, setting 'CLD' by means of a call to CPSETI will probably be inappropriate, since, by default, DASHPACK expects underscores, rather than apostrophes, to represent gaps in the dash pattern.)

There are three special elements in the parameter array 'CLD', corresponding to 'PAI' values of -1, -2, and -3; the first specifies a dash pattern for the edge of the grid, the second a dash pattern for the edge of any area filled with special values, and the third a dash pattern for the edge of any area in which the mapping routine CPMPXY returns the “out of range” value.

When CONPACK picks the contour levels, the default value supplied for each associated dash pattern is the character constant '$$$$$$$$$$$$$$$$'. This is also the default value for each of the three special elements.

4.9.20 'CLL' - Contour Line Line Width - Real Array

Each element of the parameter array 'CLL' specifies the line width used to draw contour lines at the level specified by the associated element of the contour level array 'CLV'. Each is expressed as a multiple of the “normal” line width; values less than or equal to zero imply that line width should not be set.

There are three special elements in the parameter array 'CLL', corresponding to 'PAI' values of -1, -2, and -3; the first specifies a line width for the edge of the grid, the second a line width for the edge of any area filled with special values, and the third a line width for the edge of any area in which the mapping routine CPMPXY returns the “out of range” value.

When CONPACK picks the contour levels, the default value supplied for each associated line width is 0. This is also the default value for each of the three special elements.

4.9.21 'CLS' - Contour Level Selection Flag - Integer

This parameter must be set prior to the call to CPRECT, CPSPS1, or CPSPS2 which initiates the process of drawing a particular contour plot; it specifies how contour levels are to be selected, as follows:

- If 'CLS' has the value “0”, CONPACK will not pick contour levels at all; the current values of the parameters 'NCL', 'CLV', and associated arrays will not be changed. They will thus retain the values chosen by CONPACK during a previous call or the values supplied by the user.

- If 'CLS' has a negative value of the form “-n”, CONPACK will generate “n” contour levels, splitting the range from the minimum field value to the maximum field value into “n+1” equal intervals.
- If 'CLS' has a positive value of the form “+n” and 'CIS' is less than or equal to zero, CONPACK will use values of the form “bk”, where “b” is a “base” value chosen by CONPACK and “k” is an integer. The base value “b” will be a “nice” value (as defined by the contents of the parameter array 'CIT'), chosen in such a way as to give at least “n” contour levels (with the default contents of the array 'CIT', you may get as many as “2n” levels).
- If 'CLS' has a positive value of the form “+n” and 'CIS' is greater than zero and 'CMN' is greater than 'CMX', CONPACK will use values of the form “'CIS'*k”, where “k” is an integer.
- If 'CLS' has a positive value of the form “+n” and 'CIS' is greater than zero and 'CMN' is less than or equal to 'CMX', CONPACK will use values of the form “'CMN'+'CIS'*k” which are greater than or equal to 'CMN' and less than or equal to 'CMX', where “k” is an integer.

The default value of 'CLS' is 16.

4.9.22 'CLU' - Contour Level Use Flags - Integer Array

Each element of the parameter array 'CLU' indicates how the associated contour level, in the parameter array 'CLV', is to be used. The value 0 means that no contour line is to be drawn at the associated level, the value 1 that the line is to be drawn without labels, the value 2 that the labels are to be drawn, but not the line, and the value 3 that both the line and the labels are to be drawn.

There are three special elements in the parameter array 'CLU', corresponding to 'PAI' values of -1, -2, and -3; the first specifies a flag for the edge of the grid, the second a flag for the edge of any area filled with special values, and the third a flag for the edge of any area in which the mapping routine CPMPXY returns the “out of range” value. In each case, if the flag is zero, the associated edge is not drawn; otherwise, the associated edge is drawn.

When CONPACK chooses the contour levels, the associated elements of 'CLU' are given one of the two values 1 or 3, depending on whether the line is to be labelled or not. The default values of the special elements are all zeroes.

4.9.23 'CLV' - Contour Level Values - Real Array

Each of the first 'NCL' elements of the parameter array 'CLV' is a contour level for which something is to be done (the drawing of contour lines, the drawing of contour labels, and/or the addition of contour lines to an area map).

Only elements 1 through 'NCL' may be accessed via the parameter-setting routines. Thus, code to set the contour levels and associated quantities must begin with a call to set 'NCL'.

A side effect of setting the element numbered 'PAI' of 'CLV' is that the associated element number 'PAI' in each of the parameter arrays 'AIA', 'AIB', 'CLC', 'CLD', 'CLL', 'CLU', 'LLC', and 'LLT' is also given a default value. These values are as follows:

```
PARAMETER  DEFAULT VALUE
---------  -----------------
  'AIA'    'PAI'+1
  'AIB'    'PAI'
  'CLC'    -1
  'CLD'    '$$$$$$$$$$$$$$$$'
  'CLL'     0
  'CLU'     1
  'LLC'    -1
  'LLT'    ' ' (a single blank)
```

Thus, in code to set contour levels and associated quantities, each contour level must be set before the quantities associated with it.

A side effect of setting element number 1 of 'CLV' is that the parameter 'CIU', which indicates what contour interval was used, is zeroed. It is assumed that this will only happen when the user is providing all the contour levels, in which case the concept of the “contour interval” may or not be well defined. See the description of 'CIU' for more information.

4.9.24 'CMN' - Contour Minimum - Real

When 'CLS' is greater than zero and 'CIS' is also greater than zero, if 'CMN' is less than or equal to 'CMX', then the contour levels used will be of the form 'CMN', 'CMN'+'CIS', 'CMN'+2*'CIS', ... , 'CMN'+n*'CIS', where “n” is the largest integer such that 'CMN'+n*'CIS' is less than or equal to 'CMX'. The labelled levels will be those for which “n” is a multiple of 'LIS'.

The default values of 'CMN' and 'CMX' are 1 and 0, respectively.

4.9.25 'CMX' - Contour Maximum - Real

See the description of 'CMN', above.

4.9.26 'CTM' - Character Temporary - Character

The parameter name 'CTM' refers to a temporary character buffer in CONPACK; the name may be used in the routine CPCHCL to get the dash pattern for the current line and in the routines CPCHHL, CPCHIL, and CPCHLL to get the text of the label being written or to change it.

The parameter 'CTM' has no meaningful default value.

4.9.27 'CWM' - Character Width Multiplier - Real

All character size parameters are multiplied by 'CWM'; this makes it easy to scale the sizes up or down simultaneously. Parameters affected by this are 'CFS', 'CFW', 'DPS', 'DPV', 'HLS', 'HLW', 'ILS', 'ILW', 'LLS', and 'LLW'.

The default value of 'CWM' is 1.

4.9.28 'DPS' - Dash Pattern Size - Real

The parameter 'DPS' specifies the nominal size (width) of a character in a label in a dash pattern, as a fraction of the width of the viewport. This nominal size is multiplied by 'CWM'.

The default value of 'DPS' is .010.

4.9.29 'DPU' - Dash Pattern Use Flag - Integer

If 'DPU' is zero, no dash patterns are to be used and contour lines are to be drawn using calls to CURVE.

If 'DPU' is non-zero, dash patterns are to be used: if 'DPU' is greater than zero, contour lines are to be drawn using calls to the DASHCHAR routine CURVED, but if 'DPU' is less than zero, contour lines are to be drawn using calls to the DASHPACK routine DPCURV.

When the label positioning flag ABS('LLP') = 1, contour lines are caused to be labelled by using a dash pattern formed by concatenating “n” repetitions of the appropriate element of 'CLD' (the nominal dash pattern for the line) and the appropriate element of 'LLT' (the numeric label for the line); in this case, the absolute value of 'DPU' specifies the value of “n”. When ‘DPU' is less than zero and 'NET' = 1, numeric labels generated by CONPACK will contain function codes that are meaningful to PLCHHQ; if it is desired to make these numeric labels bend with the curve they label, “break” characters must be inserted; the example “cpex12” shows how to do this.

The default value of 'DPU' is 3.

4.9.30 'DPV' - Dash Pattern Vector Length - Real

The parameter 'DPV' specifies the nominal length of the solid vectors and the gap vectors resulting from the use of a dash pattern, as a fraction of the width of the viewport. This nominal length is multiplied by 'CWM'.

The default value of 'DPV' is .005.

4.9.31 'GIC' - Group Identifier for Contour Lines - Integer

The parameter 'GIC' specifies the group identifier for contour lines added to an area map by the routine CPCLAM.

The default value of 'GIC' is 3.

4.9.32 'GIL' - Group Identifier for Label Boxes - Integer

The parameter 'GIL' specifies the group identifier for label boxes added to an area map by the routine CPLBAM.

The default value of 'GIL' is 3.

4.9.33 'GIS' - Group Identifier for Strips - Integer

The parameter 'GIS' specifies the group identifier for a group of edges added to an area map by the routine CPCLAM to create a set of vertical strips. This is done only if the parameter 'NVS' (which is described later in this section) is non-zero.

The default value of 'GIS' is 4.

4.9.34 'HCF' - Hachuring Flag - Integer

The parameter 'HCF' is the hachuring flag, with one of the following values:

- If 'HCF' = 0, hachuring is turned off (the default).
- If 'HCF' = 1, all contours will be hachured.
- If 'HCF' = 2, closed contours will be hachured only if the interior of the contour is “downhill”. Open contours will be unconditionally hachured.
- If 'HCF' = 3, closed contours will be hachured only if the interior of the contour is “downhill”. Open contours will be unconditionally unhachured.
- If 'HCF' = 4, closed contours will be hachured only if the interior of the contour is “downhill”. Open contours will be hachured only if the “interior” of the contour is “downhill”, where “interior” is defined by computing the total change in direction along the contour. If that total is positive, the “interior” is to the left; if it is negative, the “interior” is to the right.

Negative values of 'HCF' may also be used. Each has the same effect as the corresponding positive value, except that the word “downhill” in the definition above is replaced by the word “uphill”.

The default value of 'HCF' is 0.

The user will be expected to increase the value of the internal parameter 'RWC' (“Real Workspace for Contours”) from its default 100 to a value large enough to accommodate any contour that can arise from his/her data field; this may require increasing the size of the real workspace array. (This is so that the hachuring routine is assured of seeing all of each contour line. Otherwise, it can't tell which way the "inside" and the "outside" of the closed contours are.)

Closed contours that are broken into pieces that appear to be open contours (either because portions of the closed contour disappear as a result of the mapping implied by non-zero values of the parameter 'MAP' and the out-of-range value 'ORV', or because they run through special-value areas, as defined by the internal parameter 'SPV', or because the user has forgotten to increase the value of 'RWC', as described in the previous paragraph), will be hachured. The rationale for this is that no real confusion can arise from hachuring a contour line which shouldn't be, only from not hachuring one that should be.

4.9.35 'HCL' - Hachure Length - Real

The parameter 'HCL' specifies the hachure length, stated as a fraction of the width of the viewport. A positive value implies the use of hachures on the downslope side of the contour. A negative value implies the use of hachures on the upslope side of the contour.

The default value of 'HCL' is .004.

4.9.36 'HCS' - Hachure Spacing - Real

The parameter 'HCS' specifies the hachure spacing, stated as a fraction of the width of the viewport. This is the distance between one hachure and the next along a contour line. Values outside the range [.0001,10.] are forced to the nearer end of that range.

The default value of 'HCS' is .01.

4.9.37 'HIC' - High Label Color Index - Integer

The parameter 'HIC' is used in determining the color index for high labels. See the description of 'HLC', below.

4.9.38 'HIT' - High Label Text String - Character

The parameter 'HIT' specifies the text string to be used in labelling a high. See the description of 'HLT', below.

4.9.39 'HLA' - High/Low Label Angle - Real

The parameter 'HLA' specifies the angle (in degrees counterclockwise from a vector pointing to the right) at which high and low labels are to be written.

The default value of 'HLA' is 0.

4.9.40 'HLB' - High/Low Label Box Flag - Integer

If 'HLB' is zero, high and low labels will not be boxed at all. The value 1 implies that the perimeter of the box is to be drawn (in the same color as the label) after the label is drawn. The value 2 implies that the box is to be filled (in the color specified by 'LBC') before the label is drawn. The value 3 implies both of the above.

The default value of 'HLB' is 0.

4.9.41 'HLC' - High/Low Label Color Index - Integer

The parameter 'HLC' is used in determining the color index for high and low labels.

The color index for high labels is determined in this manner: If 'HIC' is greater than or equal to zero, 'HIC' is used as the color index. If 'HIC' is less than zero, but 'HLC' is greater than or equal to zero, 'HLC' is used as the color index. If both 'HIC' and 'HLC' are less than zero, the current text color index is used. If a box is drawn around the label, it is made the same color as the label itself.

The color index for low labels is determined similarly: If 'LOC' is greater than or equal to zero, 'LOC' is used as the color index. If 'LOC' is less than zero, but 'HLC' is greater than or equal to zero, 'HLC' is used as the color index. If both 'LOC' and 'HLC' are less than zero, the current text color index is used. If a box is drawn around the label, it is made the same color as the label itself.

To set the color index of all high and low labels, simply supply the desired value for 'HLC'. To have highs and low labels which are colored differently, set 'HIC' and 'LOC'.

The default values of 'HLC', 'HIC', and 'LOC' are all -1's.

4.9.42 'HLL' - High/Low Line Width - Real

If 'HLL' has a value less than or equal to zero, line width will not be set before drawing boxes around high and low labels. If 'HLL' has a value greater than zero, it specifies the desired width, as a multiple of the "normal" line width.

The default value of 'HLL' is 0.

4.9.43 'HLO' - High/Low Label Overlap Flag - Integer

The value of 'HLO' says what is to be done about the problem of high and low labels overlapping other objects. The value 0 implies that the problem will be ignored; high and low labels will not be checked for overlap with anything else. Adding 1 to the value of 'HLO' implies the omission of high and low labels which overlap the informational label. Adding 2 implies the omission of high and low labels which overlap other high and low labels found before it. Adding 4 implies the omission of high and low labels which overlap the edges of the viewport, while adding 8 implies that high and low labels which overlap the edges of the viewport should be moved inward by just enough to take care of the problem. If you add both 4 and 8, the effect will be as if you had added 4 alone.

The default value of 'HLO' is 3 (1 + 2).

4.9.44 'HLS' - High/Low Label Size - Real

The parameter 'HLS' specifies the nominal size (width) of a character in a high or low label, as a fraction of the width of the viewport. This nominal size is multiplied by 'CWM'.

The default value of 'HLS' is .012.

4.9.45 'HLT' - High/Low Label Text Strings - Character

The character strings used to label highs and lows may be specified individually, by setting 'HIT' and 'LOT', or together, by setting 'HLT'. If 'HLT' is set, and there are no apostrophes in the given character string, both 'HIT' and 'LOT' will be set equal to it and it will therefore be used as the label for both highs and lows. If there are apostrophes in the string, what precedes the first one will be used as the value of 'HIT' (the label for a high) and what follows it will be used as the value of 'LOT' (the label for a low).

Remember that, in FORTRAN, an apostrophe in a string which is delimited by apostrophes is represented by two apostrophes.

The substring ZDV may be used to represent the numeric value of the high or the low, divided by the current scale factor; the substring $ZDVU$ may be used to represent the unscaled value.

Some examples:

```
FORTRAN STRING                 HIGH LABEL    LOW LABEL
------------------------       ------------- -------------

'H''L'                             H             L

'HI''LO'                           HI            LO

'$ZDV$'                            1.362         .764

'H($ZDV$)''L($ZDV$)'               H(1.362)      L(.764)

'H:B:$ZDV$:E:''L:B:$ZDV$:E:'       H             L
                                    1.362         .764
```

Note that, in the final example, the subscripting capability of the utility PLOTCHAR is used. The terminating function code “E” ensures that the “H” or the “L” will be centered on the high or low; to center the whole thing, either remove the 'E's or change them to 'N's.

Neither of the character strings 'HIT' and 'LOT' may contain more than 32 characters.

If 'HIT' is blank, highs will not be labelled. If 'LOT' is blank, lows will not be labelled.

The default value for 'HIT' is 'H:B:ZDV:E:' and the default value of 'LOT' is 'L:B:ZDV:E:', as shown in the final example above.

4.9.46 'HLW' - High/Low Label White Space Width - Real

The parameter 'HLW' specifies the nominal width of white space to be left around a high or low label, as a fraction of the width of the viewport. This nominal width is multiplied by 'CWM'.

The default value of 'HLW' is .005.

4.9.47 'HLX' - High/Low Search Radius in X - Integer

If 'HLX' is greater than zero, it specifies the half-width of the index-value neighborhood used in searching the contour field for highs and lows. If 'HLX' is less than or equal to zero, CONPACK picks a reasonable value to use (approximately 1/8 of 'ZDM', but not less than 2 nor greater than 15).

As an example, if 'HLX' = 3 and 'HLY' = 4, then the values in ZDAT examined to determine if (I,J) is a high or a low are those having indices (K,L), where either K is not equal to I or L is not equal to J, K is between MAX(1,I-3) and MIN('ZDM',I+3), inclusive, and L is between MAX(1,J-4) and MIN('ZDN',J+4), inclusive.

The default value of 'HLX' is 0.

4.9.48 'HLY' - High/Low Search Radius in Y - Integer

If 'HLY' is greater than zero, it specifies the half-height of the index-value neighborhood used in searching the contour field for highs and lows. If 'HLY' is less than or equal to zero, CONPACK picks a reasonable value to use (approximately 1/8 of 'ZDN', but not less than 2 nor greater than 15).

For an example, see the description of 'HLX', above.

The default value of 'HLY' is 0.

4.9.49 'ILA' - Informational Label Angle - Real

The parameter 'ILA' specifies the angle (in degrees counterclockwise from a vector pointing to the right) at which the informational label is to be written.

The default value of 'ILA' is 0.

4.9.50 'ILB' - Informational Label Box Flag - Integer

If 'ILB' is zero, the informational label will not be boxed at all. The value 1 implies that the perimeter of the box is to be drawn (in the same color as the label) after the label is drawn.

The value 2 implies that the box is to be filled (in the color specified by 'LBC') before the label is drawn. The value 3 implies both of the above.

The default value of 'ILB' is 0.

4.9.51 'ILC' - Informational Label Color Index - Integer

If 'ILC' is less than zero, the informational label and the box, if any, around it, will be drawn in the color specified by the current text color index; if 'ILC' is greater than or equal to zero, then it specifies the desired color index for the label and the box. If a box is drawn around the label, it is made the same color as the label itself.

The default value of 'ILC' is -1.

4.9.52 'ILL' - Informational Label Line Width - Real

If 'ILL' has a value less than or equal to zero, line width will not be set before drawing a box around the informational label. If 'ILL' has a value greater than zero, it specifies the desired width, as a multiple of the “normal” line width.

The default value of 'ILL' is 0.

4.9.53 'ILP' - Informational Label Positioning Flag - Integer

The parameter 'ILP' says how the informational label is to be positioned. There are nine possible values, each of which specifies a point of the label box which is to lie on the point defined by 'ILX' and 'ILY': the value -4 implies the lower left-hand corner of the label box, -3 implies the center of the bottom of the box, -2 the lower right-hand corner of the box, -1 the center of the left edge of the box, 0 the center of the box, +1 the center of the right edge of the box, +2 the upper left-hand corner of the box, +3 the center of the top edge of the box, and +4 the upper right-hand corner of the box. Left, right, bottom, and top are defined from the viewpoint of someone reading the label right-side up.

The default value of 'ILP' is 4, so the upper right-hand corner of the box will be placed on the point ('ILX','ILY').

4.9.54 'ILS' - Informational Label Size - Real

The parameter 'ILS' specifies the nominal size (width) of a character in the informational label, as a fraction of the width of the viewport. This nominal size is multiplied by 'CWM'.

The default value of 'ILS' is .012.

4.9.55 'ILT' - Informational Label Text String - Character

The parameter 'ILT' is a string of 128 or fewer characters, specifying the text of the informational label. The following substrings will be replaced by a numeric value:

```
$CIU$ - THE CONTOUR INTERVAL USED.
$CMN$ - THE MINIMUM CONTOUR LEVEL.
$CMX$ - THE MAXIMUM CONTOUR LEVEL.
$SFU$ - THE CURRENT SCALE FACTOR.
$ZMN$ - THE MINIMUM VALUE IN THE DATA ARRAY.
$ZMX$ - THE MAXIMUM VALUE IN THE DATA ARRAY.
```

In each case except SFU, the given value will have been divided by the current scale factor. A “U” may be inserted just before the final “$” (as in '$CIUU$', '$CMNU$', etc.) to request the use of an unscaled value.

The value with which CIU is replaced will only be correct if CONPACK itself has chosen the contour levels; otherwise, it may be necessary for the user to set the value of 'CIU' (which see, above).

If 'ILT' is given the value ' ' (a single blank), there will be no informational label.

The default value of 'ILT' is 'CONTOUR FROM CMN TO CMX BY CIU'.

4.9.56 'ILW' - Informational Label White Space Width - Real

The parameter 'ILW' specifies the nominal width of white space to be left around the informational label, as a fraction of the width of the viewport. This nominal width is multiplied by 'CWM'.

The default value of 'ILW' is .005.

4.9.57 'ILX' - Informational Label X Coordinate - Real

The parameter 'ILX' specifies the X coordinate of the basepoint of the informational label. The given value is mapped linearly onto the viewport; 0 refers to the left edge of the viewport and 1 to the right edge of the viewport. Values less than 0 or greater than 1 may be used.

The default value of 'ILX' is .98.

4.9.58 'ILY' - Informational Label Y Coordinate - Real

The parameter 'ILY' specifies the Y coordinate of the basepoint of the informational label. The given value is mapped linearly onto the viewport; 0 refers to the bottom edge of the viewport and 1 to the top edge of the viewport. Values less than 0 or greater than 1 may be used.

The default value of 'ILY' is -.02.

4.9.59 'IWM' - Integer Workspace for Masking - Integer

The parameter 'IWM' specifies the amount of integer workspace to be allotted for use by CPCLDM, which draws contour lines masked by an area map, in calls to the routine

ARDRLN, in the package AREAS. Assume a parameter value “n”; the space used will be “2n” (“n” for the array IAI and “n” for the array IAG, in calls to ARDRLN). The value “n” must be greater than or equal to the number of group identifiers used in generating the area map.

The default value of 'IWM' is 10.

4.9.60 'IWU' - Integer Workspace Usage - Integer

The parameter 'IWU' is intended for retrieval only. It is zeroed by the call to CPRECT, CPSPS1, or CPSPS2. Thereafter, as CONPACK routines are called, the value of 'IWU' is updated to reflect the largest number of words of integer workspace needed at any one time. Therefore, by retrieving its value after an entire plot has been constructed, one may find out how large an integer workspace was actually required.

4.9.61 'LBC' - Label Box Color Index - Integer

If label boxes (of whatever type) are filled, the filling is done using the color index specified by 'LBC'. If 'LBC' is less than zero, the current fill area color index is used.

The default value of 'LBC' is 0, which specifies the background color.

4.9.62 'LBX' - Label Box X Coordinate - Real

Not to be set by the user. The value may be retrieved in one of the routines CPCHCF, CPCHHL, CPCHIL, or CPCHLL. It specifies the X coordinate (in the current user coordinate system) of the center of the box surrounding the label that has caused the routine to be called.

The default value of 'LBX' is 0.

4.9.63 'LBY' - Label Box Y Coordinate - Real

Not to be set by the user. The value may be retrieved in one of the routines CPCHCF, CPCHHL, CPCHIL, or CPCHLL. It specifies the Y coordinate (in the current user coordinate system) of the center of the box surrounding the label that has caused the routine to be called.

The default value of 'LBY' is 0.

4.9.64 'LIS' - Label Interval Specifier - Integer

When 'CLS' is given a positive value, indicating that CONPACK is to choose contour levels at intervals of the form “bk”, where “b” is a base value and “k” is an integer, and 'CIS' is given a positive value, indicating that it is the desired value of “b”, then 'LIS' must be set to specify the interval between labelled contour levels.

See the descriptions of the parameters 'CLS' and 'CIS'.

As an example, one might specify 'CLS'=1, 'CIS'=1/3 and 'LIS'=3 in order to get contours at values like 1/3, 2/3, 3/3, 4/3, etc., with labels at values like 1, 2, 3, etc.

The default value of 'LIS' is 5.

4.9.65 'LIT' - Label Interval Table - Integer Array

See the description of the parameter 'CIT'.

4.9.66 'LIU' - Label Interval Used - Integer

The parameter 'LIU' is for retrieval only. When CONPACK chooses the contour interval and decides that every "nth" one should be labelled, it sets 'LIU' to "n".

4.9.67 'LLA' - Line Label Angle - Real

The parameter 'LLA' specifies the angle (in degrees counterclockwise from a vector pointing to the right) at which contour line labels are to be written when ABS('LLP') is greater than or equal to 2 and 'LLO' is 0.

The default value of 'LLA' is 0.

4.9.68 'LLB' - Line Label Box Flag - Integer

If 'LLB' is zero, contour line labels drawn by CPLBDR will not be boxed at all. The value 1 implies that the perimeter of the box is to be drawn (in the same color as the label) after the label is drawn. The value 2 implies that the box is to be filled (in the color specified by 'LBC') before the label is drawn. The value 3 implies both of the above.

The default value of 'LLB' is 0.

4.9.69 'LLC' - Line Label Color Index - Integer Array

Each element of the parameter array 'LLC', if greater than or equal to zero, is the color index for labels on contour lines at the level specified by the associated element of 'CLV'. A value less than zero implies that the current text color index is to be used.

This parameter only affects line labels when ABS('LLP') = 2 or 3 and the labels are therefore drawn by CPLBDR. It does not affect line labels when ABS('LLP') = 1 and the line labels are therefore drawn by a dash package routine called by CPCLDM or CPCLDR.

The default values of the elements of 'LLC' are all -1's.

4.9.70 'LLL' - Line Label Line Width - Real

If 'LLL' has a value less than or equal to zero, line width will not be set before drawing boxes around contour line labels. If 'LLL' has a value greater than zero, it specifies the desired width, as a multiple of the “normal” line width.

The default value of 'LLL' is 0.

4.9.71 'LLO' - Line Label Orientation - Integer

The parameter 'LLO' only has effect when ABS('LLP') is greater than or equal to 2, specifying use of either the regular scheme or the penalty scheme for positioning labels on contour lines. If 'LLO' is zero, the labels are written at the angle specified by 'LLA'. If 'LLO' is non-zero, the labels are written in the local direction of the contour line.

The default value of 'LLO' is 0.

4.9.72 'LLP' - Line Label Positioning - Integer

The parameter 'LLP' says whether or not labels should be produced and, if so, how, as follows:

- If 'LLP' is 0, it says that no line labels should be produced.
- If ABS('LLP') is 1, it says that labels should be positioned along contour lines by using the old CONREC scheme of setting up a character dash pattern including the label and using a software dash package to draw the labels (which requires having 'DPU' set non-zero). This scheme has the disadvantages that one cannot control the orientation and one cannot shield the labels from having contour lines drawn through them.
- If ABS('LLP') is 2, it says that labels should be positioned at regular intervals along the line. See the descriptions of the parameters 'RC1', 'RC2', and 'RC3'.
- If ABS('LLP') is 3, it says that labels should be positioned using a penalty scheme, which gives much better results than either of the others.

When 'LLP' is 2 or 3, the 2D smoothing, if any, implied by the value of 'T2D' is suspended during label positioning, so that fewer label positions will be considered; this is quite a bit faster and the results are nearly as good as if the smoothing were done. To force smoothing, use 'LLP' = -2 or -3.

The default value of 'LLP' is 1.

4.9.73 'LLS' - Line Label Size - Real

The parameter 'LLS' specifies the nominal size (width) of a character in a contour line label written by one of the schemes used when 'LLP' is 2 or 3; the value of 'LLS' is given as a fraction of the width of the viewport. This nominal size is multiplied by 'CWM'.

The value of 'LLS' does not affect contour line labels being written by a dash package, as happens when 'LLP' has the value 1. The width of the characters used in writing those labels is determined instead by the value of the internal parameter 'DPS'.

The default value of 'LLS' is .010.

4.9.74 'LLT' - Line Label Text String - Character

For each I from 1 to 'NCL', element I of the parameter array 'LLT' is a string of 64 or fewer characters, to be used as a label for the contour level specified by the Ith element of 'CLV'. Since the character string will be plotted using the routine PLCHHQ, in the package PLOTCHAR, it may contain colon-enclosed “function codes” to do things like create superscripts.

If the elements of this array are not supplied by the user, they will be filled in by CONPACK itself.

4.9.75 'LLW' - Line Label White Space - Real

The parameter 'LLW' specifies the nominal width of white space to be left around a contour line label, as a fraction of the width of the viewport. This nominal width is multiplied by 'CWM'.

The default value of 'LLW' is .005.

4.9.76 'LOC' - Low Label Color Index - Integer

The parameter 'LOC' is used in determining the color index for low labels. See the description of 'HLC', above.

4.9.77 'LOT' - Low Label Text String - Character

The parameter 'LOT' specifies the text string to be used in labelling a low. See 'HLT', above.

4.9.78 'MAP' - Mapping Flag - Integer

If 'MAP' is zero, it says that the x and y coordinates used to create the contour map are not to be transformed by the user-replaceable subroutine CPMPXY. If 'MAP' is non-zero, it says that x and y coordinates are to be so transformed. The default version of CPMPXY provides two useful mappings:

- If the first subscript of the data array is a linear function of the longitude and the second is a linear function of the latitude, then one can transform all graphics output onto a map background created by calls to routines in the utility package EZMAP just by setting 'MAP' = 1, 'XC1' = minimum longitude, 'XCM' = maximum longitude, 'YC1' =

minimum latitude, and 'YCN' = maximum latitude. Also, the parameter 'SET' must be given the value 0 in order to prevent CONPACK from calling SET and thereby overriding the call done by EZMAP.

- If the first subscript of the data array is a linear function of rho and the second is a linear function of theta, where rho and theta are polar coordinates, then to map all graphics output properly, one may set 'MAP' = 2, 'XC1' = minimum rho, 'XCM' = maximum rho, 'YC1' = minimum theta, and 'YCN' = maximum theta. In this case, one must either use 'SET' = 0 and do an appropriate SET call or use 'SET' = 1 and give the parameters 'WDB', 'WDL', 'WDR', and 'WDT' values consistent with the mapped values of X and Y, which will all be of the form “rho*cos(theta)” and “rho*sin(theta)”, respectively.

Using any other non-zero value of 'MAP' will result in the identity mapping.

Of course, one can replace the routine CPMPXY and build as many different mappings into it as desired. See the description of CPMPXY.

The default value of 'MAP' is 0.

4.9.79 'NCL' - Number of Contour Levels - Integer

If CONPACK is to pick contour levels (see the description of the parameter 'CLS') then the initial call to CPRECT, CPSPS1, or CPSPS2 causes 'NCL' to be zeroed. Subsequently, during the first call to a CONPACK routine requiring contour levels to have been chosen, 'NCL' will be set as part of the process of choosing them. If the user elects to choose the contour levels, the first parameter which must be set to do this is 'NCL'.

The parameter 'NCL' has no meaningful default value.

4.9.80 'NEL' - Numeric Exponent Length - Integer

Giving 'NEL' a value less than or equal to zero says that exponents in numeric labels should be written in the shortest possible form; plus signs are omitted and the exponent magnitude is written with no leading zeroes. A value “n” which is greater than zero indicates that all exponents should be written with a sign (+ or -) and that the exponent magnitude should be padded with leading zeroes to a length of n characters.

The default value of 'NEL' is 0.

4.9.81 'NET' - Numeric Exponent Type - Integer

The parameter 'NET' says what characters are to be used between the mantissa of a numeric label and the exponent. The value 0 implies the use of an E, as in FORTRAN “E format”, the value 1 implies the use of function codes, as expected by the utility PLOTCHAR, to generate “x 10^n”, where n is a superscript exponent, and the value 2 implies the use of “x10**”.

The default value of 'NET' is 1.

4.9.82 'NEU' - Numeric Exponent Use Flag - Integer

Giving 'NEU' a value less than or equal to zero forces the use of the exponential form in all numeric labels. A positive value "n" indicates that the form without an exponent should be used as long as it requires no more than n characters; otherwise the form requiring the fewest characters should be used.

The default value of 'NEU' is 5.

4.9.83 'NLS' - Numeric Leftmost Significant Digit Flag - Integer

Giving 'NLS' the value zero says that the leftmost non-zero digit of a number represented by a numeric label is to be considered its first significant digit. A non-zero value says that the digit in the same digit position as the leftmost non-zero digit of the largest number (in absolute value) in the data field being contoured is to be considered the leftmost significant digit. This tends to make the numeric labels more nearly consistent with one another. Consider the following example, using three significant digits:

```
USING 'NLS'=0: .500 1.00 1.50 ... 9.50 10.5 ...
USING 'NLS'=1: .5   1.0  1.5  ... 9.5  10.5 ...
```

The default value of 'NLS' is 1.

4.9.84 'NLZ' - Numeric Leading Zero Flag - Integer

Giving 'NLZ' a non-zero value says that a zero is to placed before any numeric label which would otherwise begin with a decimal point (use "0.345", rather than ".345").

The default value of 'NLZ' is 0.

4.9.85 'NOF' - Numeric Omission Flags - Integer

The parameter 'NOF' says what parts of a numeric label may be omitted. The value 0 says that no part may be omitted. Add a 4 to indicate that a leading "1" or "1." which is unnecessary (as in "1×10^{13}") may be omitted, a 2 to indicate that a trailing decimal point (as in "13.") may be omitted, and a 1 to indicate that trailing zeroes (as in "46.200") may be omitted.

Contour line labels generated by CONPACK and values in the informational label which are known to have been rounded to "nice" values (like 'CIU', 'CMN', and 'CMX') will have trailing zeroes trimmed in any case, no matter what the value of 'NOF' is.

The default value of 'NOF' is 6 (4 + 2).

4.9.86 'NSD' - Number of Significant Digits - Integer

The parameter 'NSD' specifies the maximum number of significant digits to be used in numeric labels representing contour field values. A negative value "-n" indicates that n significant digits should be used. A positive value “n” indicates that “m+n” digits should be used, where “m” is the number of digits that are the same for all values in the contour field. (For example, if the minimum value is “1123.6” and the maximum value is “1125.9”, then the value of “m” is 3.)

The default value of 'NSD' is 4.

4.9.87 'NVS' - Number of Vertical Strips - Integer

When the parameter 'NVS' is non-zero, an extra group of edges, with group identifier 'GIS', is added to the area map by the routine CPCLAM. These edges include the boundary of the viewport and enough vertical lines to break the area occupied by the viewport up into 'NVS' vertical strips. The object of this is to break up the contour bands which are to be filled into smaller and simpler pieces; this may be necessary if the graphics device in use limits the number of points which may be used to define a polygon to be filled. The area identifier for the outside of the viewport is -1; all other area identifiers used are 0’s.

The default value of 'NVS' is 1.

4.9.88 'ORV' - Out-of-Range Value - Real

If 'ORV' is non-zero, it specifies an out-of-range value, to be used as the value of X and Y coordinates returned by the mapping routine CPMPXY to say that a point is out-of range (invisible) under the current mapping.

When 'MAP' = 1, a good value for 'ORV' is 1.E12, since that value is returned by EZMAP routines to signal such a point.

The default value of 'ORV' is 0.

4.9.89 'PAI' - Parameter Array Index - Integer

The value of 'PAI' must be set before calling CPGETC, CPGETI, CPGETR, CPSETC, CPSETI, or CPSETR to access any parameter which is an array; it indicates which element of the array is meant. For example, to set the 10th contour level to zero, use code like this:

```
CALL CPSETI ('PAI - PARAMETER ARRAY INDEX',10)
CALL CPSETR ('CLV - CONTOUR LEVEL',0.)
```

The default value of 'PAI' is 0.

4.9.90 'PC1' - Penalty Scheme Constant 1 - Real

The parameter 'PC1' is one of the constants used in the penalty scheme for positioning labels. The largest gradient allowed at the position of a label will be GRAV + 'PC1' * GRSD, where GRAV is the average gradient and GRSD is the standard deviation of the gradients.

The default value of 'PC1' is 1.

4.9.91 'PC2' - Penalty Scheme Constant 2 - Real

The parameter 'PC2' is one of the constants used in the penalty scheme for positioning labels. It specifies the maximum (estimated) number of contour bands allowed to cross a label.

The default value of 'PC2' is 5.

4.9.92 'PC3' - Penalty Scheme Constant 3 - Real

The parameter 'PC3' is one of the constants used in the penalty scheme for positioning labels. It specifies, in degrees, the maximum cumulative change in direction to be allowed along that portion of the contour line covered by a circle centered on a label and having a radius equal to half the width of the label.

The default value of 'PC3' is 60.

4.9.93 'PC4' - Penalty Scheme Constant 4 - Real

The parameter 'PC4' is one of the constants used in the penalty scheme for positioning labels. It specifies the “optimal distance” in the term in the penalty function which attempts to force labels to be at an optimal distance from each other.

The default value of 'PC4' is .05.

4.9.94 'PC5' - Penalty Scheme Constant 5 - Real

The parameter 'PC5' is one of the constants used in the penalty scheme for positioning labels. It specifies the “folding distance” in the term in the penalty function which attempts to force labels to be at an optimal distance from each other.

The default value of 'PC5' is .15.

4.9.95 'PC6' - Penalty Scheme Constant 6 - Real

The parameter 'PC6' is one of the constants used in the penalty scheme for positioning labels. It specifies the minimum distance to be allowed between any two labels on the same contour line, as a fraction of the width of the viewport.

The default value of 'PC6' is .30.

4.9.96 'PIC' - Point Interpolation Flag for Contours - Integer

The parameter 'PIC' specifies the number of points to interpolate between each pair of points defining a segment of a contour line, prior to any mapping implied by the parameter 'MAP'. It is intended that a non-zero value should normally be used only if 'MAP' is non-zero, which turns mapping on, and 'T2D' is zero, which turns the 2D smoother off; the intent is to map straight-line segments of contour lines more nearly correctly into curved-line segments on a background (one drawn by EZMAP, for example). If the 2D smoother is turned on, the additional points will be used and the smoothed curve will be constrained to pass through them; this may be useful.

A negative value of 'PIC' causes ABS('PIC') points to be interpolated, but the interpolated points are not, in general, used to draw the line segment; the object, in this case, is simply to do a finer search for changes in visibility (out-of-range state, as defined by values of 'ORV' returned by the routine CPMPXY) along the segment.

See also the description of the parameter 'PIT', which can be used to cause the interpolation of points on segments of contour lines only in regions where the mapping causes great distortion.

See the example “cpex11”, which illustrates the use of 'PIC', 'PIE', and 'PIT'.

The default value of 'PIC' is 0.

4.9.97 'PIE' - Point Interpolation Flag for Edges - Integer

The parameter 'PIE' specifies the number of points to interpolate between each pair of points defining a segment of an “edge” (the edge of the grid, the edge of a special-value area, or the edge of an out-of-range area). It is intended that a non-zero value should normally be used only if 'MAP' is non-zero, which turns mapping on; the intent is to map straight-line segments of edge lines more nearly correctly into curved-line segments on a background (one drawn by EZMAP, for example).

A negative value of 'PIE' causes ABS('PIE') points to be interpolated, but the interpolated points are not, in general, used to draw the line segment; the object, in this case, is simply to do a finer search for changes in out-of-range state (visibility) along the segment. (The edges of out-of-range areas, however, are drawn using all such interpolated points.)

Using too large an (absolute) value of 'PIE' will cause the tracing of the edges of out-of-range areas to be very time-consuming, because the number of points to be examined is 'ZDM' x 'ZDN' x 'PIE' x 'PIE'.

See also the description of the parameter 'PIT', which can be used to cause the interpolation of points on segments of edges only in regions where the mapping causes great distortion.

See the example “cpex11”, which illustrates the use of 'PIC', 'PIE', and 'PIT'.

The default value of 'PIE' is 0.

4.9.98 'PIT' - Point Interpolation Threshold - Real

When 'MAP' is non-zero, the parameter 'PIT' may also be set non-zero to improve the accuracy with which contour lines and edge lines are mapped. As the points defining such a line are mapped (using CPMPXY), CONPACK looks for a jump in the mapped X coordinates of more than 'PIT' times the width of the current user-coordinate-system window or a jump in the mapped Y coordinates of more than 'PIT' times the height of the current user-coordinate-system window. When such a jump is seen, additional points are interpolated and mapped to reduce all such jumps below the threshold value. This parameter becomes particularly important when using some of the EZMAP projections that project the entire globe to the interior of a circle, in which case the points defining a segment of a contour line can map to points so far apart that the mapped contour lines cross one another.

The use of the internal parameter 'PIT' does not entirely replace the use of 'PIC' and 'PIE'; indeed, there are situations in which both can be used to good advantage, particularly when negative values of 'PIC' and 'PIE' are used to improve the resolution of the search for changes in out-of-range state along the line segments.

See the example “cpex11”, which illustrates the use of 'PIC', 'PIE', and 'PIT'.

The default value of 'PIT' is 0. It’s probably not a good idea to use values less than about .001 or greater than about .1.

4.9.99 'PW1' - Penalty Scheme Weight 1 - Real

The parameter 'PW1' specifies the weight for the gradient term in the penalty function.

The default value of 'PW1' is 2.

4.9.100 'PW2' - Penalty Scheme Weight 2 - Real

The parameter 'PW2' specifies the weight for the number-of-contours term in the penalty function.

The default value of 'PW2' is 0.

4.9.101 'PW3' - Penalty Scheme Weight 3 - Real

The parameter 'PW3' specifies the weight for the change-in-direction term in the penalty function.

The default value of 'PW3' is 1.

4.9.102 'PW4' - Penalty Scheme Weight 4 - Real

The parameter 'PW4' specifies the weight for the optimum-distance term in the penalty function.

The default value of 'PW4' is 1.

4.9.103 'RC1' - Regular Scheme Constant 1 - Real

The parameter 'RC1' specifies the desired distance from the beginning of a contour line to the first label on that line when they are positioned using the “regular” scheme. The nth label on each labelled contour line will be at a distance 'RC1' + 'RC2' x (n-1) + 'RC3' x Rn units (in the fractional coordinate system) from the beginning of the line, where “Rn” is a random number between -1 and 1.

The default value of 'RC1' is .25.

4.9.104 'RC2' - Regular Scheme Constant 2 - Real

The parameter 'RC2' specifies the desired nominal distance between labels when they are positioned using the “regular” scheme. See the description of 'RC1', above.

The default value of 'RC2' is .25.

4.9.105 'RC3' - Regular Scheme Constant 3 - Real

The parameter 'RC3' specifies the desired maximum variation in the distance between labels when they are positioned using the regular scheme. See the description of 'RC1', above.

The default value of 'RC3' is .05.

4.9.106 'RWC' - Real Workspace for Contours - Integer

The parameter 'RWC' specifies the amount of real workspace to be allotted to hold X coordinates of points defining contour lines. Assume a parameter value “n”. If no 2D smoothing is requested, the total space used will be “2n” (“n” for X coordinates and another “n” for Y coordinates). If 2D smoothing is requested, the total space used will be “7n” (“n” for X coordinates, “n” for Y coordinates, and “5n” for scratch arrays).

Normally, the value of 'RWC' is of no particular interest to the user, since the same contour lines are produced with a small value as would be produced with a larger value. There are two situations in which it becomes of more interest: 1) When the penalty scheme is used to position labels, the length of the portion of the contour line over which the penalty function is evaluated is limited by the value of 'RWC'. If 'RWC' is set too small, too many labels may be put on a given contour line and some of them may be too close to each other. 2) When hachuring has been activated (by setting the value of 'HCF' non-zero), it is important that the internal routine that does the hachuring see entire contours at once, so that it may properly decide whether a contour is open or closed and, in the latter case, where the interior of the closed contour is. In both of these cases, the solution is to increase the value of 'RWC'.

The default value of 'RWC' is 100.

4.9.107 'RWG' - Real Workspace for Gradients - Integer

The parameter 'RWG' specifies the amount of real workspace to be allotted to hold gradients which are to be computed and used in positioning labels using the penalty scheme. Using a larger value provides for a more accurate representation of the gradient field, up to the point at which it exceeds 'ZDM' x 'ZDN'.

The default value of 'RWG' is 1000.

4.9.108 'RWM' - Real Workspace for Masking - Integer

The parameter 'RWM' specifies the amount of real workspace to be allotted for use by CPCLDM, which draws contour lines masked by an area map, in calls to the routine ARDRLN, in the package AREAS. Assume a parameter value “n”; the space used will be “2n” (“n” for the X-coordinate array XCS and “n” for the Y-coordinate array YCS, in calls to ARDRLN). Any value of “n” greater than or equal to 2 will work; smaller values will cause the generation of more calls to the user routine RTPL (one of the arguments of CPCLDM).

The default value of 'RWM' is 100.

4.9.109 'RWU' - Real Workspace Usage - Integer

The parameter 'RWU' is intended for retrieval only. It is zeroed by the call to CPRECT, CPSPS1, or CPSPS2. Thereafter, as CONPACK routines are called, the value of 'RWU' is updated to reflect the largest number of words of real workspace needed at any one time. Therefore, by retrieving its value after an entire plot has been constructed, one may find out how large a real workspace was actually required.

4.9.110 'SET' - Do-SET-Call Flag - Integer

Giving 'SET' the value 0 says that no SET call is to be done by CONPACK; the value 1 says that it is to be done. In the latter case, the call is done by CPRECT, CPSPS1, or CPSPS2.

Arguments 5-8 of a SET call done by the user must be consistent with the ranges of the X and Y coordinates being used by CONPACK, as specified by the values of the parameters 'XC1', 'XCM', 'YC1', 'YCN', and 'MAP'. See the descriptions of those parameters.

The default value of 'SET' is 1.

4.9.111 'SFS' - Scale Factor Selector - Real

The scale factor is that value (usually, but not necessarily, a power of 10) by which the actual values of contour field values are to be divided to get the value of a numeric label. If 'SFS' is given a value greater than zero, that value is the scale factor to be used. If 'SFS' is given a value less than or equal to zero, it is truncated to form an integer directing CONPACK to select a scale factor, as follows:

- 0 implies that the scale factor should be selected in such a way as to reduce the ZDAT element having the largest absolute value to the range from .1 to .999...
- -1 implies that the scale factor should be selected in such a way as to reduce the ZDAT element having the largest absolute value to the range from 1. to 9.999...
- -2 implies that the scale factor should be selected in such a way as to place the decimal point in the ZDAT element having the largest absolute value after the rightmost significant digit of that value (as defined by the values of 'NSD' and 'NLS').
- -3 implies that the scale factor should be selected in such a way as to remove extra zeroes from the ends of the ZDAT element having the largest absolute value. For example, if that element were .000136, the scale factor would be 10 to the power -3; if that element were 136000 (assuming three significant digits are desired), the scale factor would be 10 to the power 3. If there are no extra zeroes on either end of the ZDAT element having the largest absolute value, the scale factor will be 1.
- -4 or less implies that the scale factor should be selected in such a way as to reduce all contour labels to integers.

The default value of 'SFS' is 1.

4.9.112 'SFU' - Scale Factor Used - Real

The parameter 'SFU' is intended for retrieval only; it gives the value of the scale factor selected for use by CONPACK.

4.9.113 'SPV' - Special Value - Real

If 'SPV' is non-zero, it specifies a “special value”, which may be used in data fields to signal missing data. No contour lines will be drawn within any grid cell with a special value at one or more of its four corners.

The default value of 'SPV' is 0.

4.9.114 'SSL' - Smoothed Segment Length - Real

The parameter 'SSL' specifies the distance between points used to draw the curves generated by 2D smoothing; it is expressed as a fraction of the width of the window in the coordinate system in which the smoothing is being done.

The default value of 'SSL' is .01.

4.9.115 'T2D' - Tension on 2-Dimensional Splines - Real

A non-zero value of 'T2D' says that 2D smoothing (using cubic splines under tension) should be done; the absolute value of 'T2D' is the desired tension. If 'T2D' is negative, smoothing will be done before the mapping, if any, requested by the flag 'MAP'; if 'T2D' is positive, smoothing will be done after the mapping.

The default value of 'T2D' is 0.

4.9.116 'T3D' - Tension on 3-Dimensional Splines - Real

The parameter 'T3D' specifies the tension on the 3D (bicubic) splines used by CPSPS1 or CPSPS2 to smooth the data being contoured.

The default value of 'T3D' is 1.

4.9.117 'VPB' - Viewport Bottom - Real

The parameter 'VPB' is only used when 'SET' is non-zero, saying that CONPACK should do the call to SET; it specifies the position of the bottom edge of the area in which the viewport is to be placed, expressed as a fraction between 0 (the bottom edge of the plotter frame) and 1 (the top edge of the plotter frame). See also the description of 'VPS'.

The default value of 'VPB' is .05.

4.9.118 'VPL' - Viewport Left - Real

The parameter 'VPL' is only used when 'SET' is non-zero, saying that CONPACK should do the call to SET; it specifies the position of the left edge of the area in which the viewport is to be placed, expressed as a fraction between 0 (the left edge of the plotter frame) and 1 (the right edge of the plotter frame). See also the description of 'VPS'.

The default value of 'VPL' is .05.

4.9.119 'VPR' - Viewport Right - Real

The parameter 'VPR' is only used when 'SET' is non-zero, saying that CONPACK should do the call to SET; it specifies the position of the right edge of the area in which the viewport is to be placed, expressed as a fraction between 0 (the left edge of the plotter frame) and 1 (the right edge of the plotter frame). See also the description of 'VPS'.

The default value of 'VPR' is .95.

4.9.120 'VPS' - Viewport Shape - Real

The parameter 'VPS' is only used when 'SET' is non-zero, saying that CONPACK should do the call to SET; it specifies the desired viewport shape, as follows:

- A negative value specifies the exact shape of the viewport; the absolute value is the ratio of the width of the viewport to its height.
- The value 0 specifies a viewport completely filling the area specified by 'VPL', 'VPR', 'VPB', and 'VPT'.
- A value “s” between 0 and 1 specifies a plot of the shape determined by the values of 'XC1', 'XCM', 'YC1', and 'YCN', reverting to the shape specified by 'VPL', 'VPR', 'VPB', and 'VPT' if the ratio of the shorter side to the longer side would be less than “s”.
- A value “s” greater than or equal to 1 specifies a plot of the shape determined by the values of 'XC1', 'XCM', YC1', and 'YCN', reverting to a square if the ratio of the longer side to the shorter side would be greater than “s”.

The viewport, whatever its final shape, is centered in, and made as large as possible in, the area specified by the parameters 'VPB', 'VPL', 'VPR', and 'VPT'.

The default value of 'VPS' is .25.

4.9.121 'VPT' - Viewport Top - Real

The parameter 'VPT' is only used when 'SET' is non-zero, saying that CONPACK should do the call to SET; it specifies the position of the top edge of the area in which the viewport is to be placed, expressed as a fraction between 0 (the bottom edge of the plotter frame) and 1 (the top edge of the plotter frame). See also the description of 'VPS'.

The default value of 'VPT' is .95.

4.9.122 'WDB' - Window Bottom - Real

When CONPACK does the call to 'SET', the parameter 'WDB' is used to determine argument number 7, the user Y coordinate at the bottom of the window. If 'WDB' is not equal to 'WDT', 'WDB' is used. If 'WDB' is equal to 'WDT', but 'YC1' is not equal to 'YCN', then 'YC1' is used. Otherwise, the value 1. is used.

The default value of 'WDB' is 0.

4.9.123 'WDL' - Window Left - Real

When CONPACK does the call to 'SET', the parameter 'WDL' is used to determine argument number 5, the user X coordinate at the left edge of the window. If 'WDL' is not equal to 'WDR', 'WDL' is used. If 'WDL' is equal to 'WDR', but 'XC1' is not equal to 'XCM', then 'XC1' is used. Otherwise, the value 1. is used.

The default value of 'WDL' is 0.

4.9.124 'WDR' - Window Right - Real

When CONPACK does the call to 'SET', the parameter 'WDR' is used to determine argument number 6, the user X coordinate at the right edge of the window. If 'WDR' is not equal to 'WDL', 'WDR' is used. If 'WDR' is equal to 'WDL', but 'XCM' is not equal to 'XC1', then 'XCM' is used. Otherwise, the value REAL('ZDM') is used.

The default value of 'WDR' is 0.

4.9.125 'WDT' - Window Top - Real

When CONPACK does the call to 'SET', the parameter 'WDB' is used to determine argument number 8, the user Y coordinate at the top of the window. If 'WDT' is not equal to 'WDB', 'WDT' is used. If 'WDT' is equal to 'WDB', but 'YCN' is not equal to 'YC1', then 'YCN' is used. Otherwise, the value REAL('ZDN') is used.

The default value of 'WDT' is 0.

4.9.126 'WSO' - Workspace Overflow Flag - Integer

The parameter 'WSO' says what to do when a real or integer workspace overflow occurs, as follows:

- The value 0 indicates that execution will terminate with a fatal-error call to SETER.
- The value 1 indicates that an error message will be written to the error file, after which execution will continue.
- The value 2 indicates that no error message will be written, after which execution will continue.
- The value 3 indicates that a recoverable-error call to SETER will be done, after which execution will continue.

When execution continues, the resulting plot will be incomplete. The values of 'IWU' and 'RWU' may be retrieved to find out how much workspace would have been used if the call on which the workspace overflow occurred had succeeded; note that, if these amounts are provided on a subsequent run, one is not assured that the workspace overflow will be averted.

The default value of 'WSO' is 1.

4.9.127 'XC1' - X Coordinate at Index 1 - Real

The parameter 'XC1' specifies the X coordinate value which corresponds to a value of 1 for the first subscript of the data array, prior to any mapping implied by a non-zero value of 'MAP'. If 'XC1' is equal to 'XCM', 1. will be used.

The default value of 'XC1' is 0.

4.9.128 'XCM' - X Coordinate at Index M - Real

The parameter 'XCM' specifies the X coordinate value which corresponds to a value of 'ZDM' for the first subscript of the data array, prior to any mapping implied by a non-zero value of 'MAP'. If 'XC1' is equal to 'XCM', REAL('ZDM') will be used.

The default value of 'XCM' is 0.

4.9.129 'YC1' - Y Coordinate at Index 1 - Real

The parameter 'YC1' specifies the Y coordinate value which corresponds to a value of 1 for the second subscript of the data array, prior to any mapping implied by a non-zero value of 'MAP'. If 'YC1' is equal to 'YCN', 1. will be used.

The default value of 'YC1' is 0.

4.9.130 'YCN' - Y Coordinate at Index N - Real

The parameter 'YCN' specifies the Y coordinate value which corresponds to a value of 'ZDN' for the second subscript of the data array, prior to any mapping implied by a non-zero value of 'MAP'. If 'YC1' is equal to 'YCN', REAL('ZDN') will be used.

The default value of 'YCN' is 0.

4.9.131 'ZD1' - ZDAT 1st Dimension - Integer

The parameter 'ZD1' specifies the first dimension of the array ZDAT, which contains the data to be contoured. If CPRECT is called, it sets 'ZD1' (the argument KZDT is the desired value). If CPSPS1 or CPSPS2 is called, it either picks a value of 'ZD1' (if 'ZDS' is non-zero) or expects the user to have done so (if 'ZDS' is zero).

The default value of 'ZD1' is 1.

4.9.132 'ZDM' - Z Data Array Dimension M - Integer

The parameter 'ZDM' specifies the first dimension of the array of data to be contoured. Its value will be less than or equal to the value of 'ZD1'. If CPRECT is called, it sets 'ZDM' (the argument MZDT is the desired value). If CPSPS1 or CPSPS2 is called, it either picks a value of 'ZDM' (if 'ZDS' is non-zero) or expects the user to have done so (if 'ZDS' is zero).

The default value of 'ZDM' is 1.

4.9.133 'ZDN' - Z Data Array Dimension N - Integer

The parameter 'ZDN' specifies the second dimension of the array of data to be contoured. If CPRECT is called, it sets 'ZDN' (the argument NZDT is the desired value). If CPSPS1 or CPSPS2 is called, it either picks a value of 'ZDN' (if 'ZDS' is non-zero) or expects the user to have done so (if 'ZDS' is zero).

The default value of 'ZDN' is 1.

4.9.134 'ZDS' - ZDAT Dimension Selector - Integer

If 'ZDS' is non-zero, CPSPS1 or CPSPS2 will select values for 'ZD1', 'ZDM', and 'ZDN'; otherwise, they will be expected to have been set by the user. Note that, if the size of the dense array is not a product of the size of the sparse array and some perfect square, the aspect ratio of the dense grid may be slightly different from that of the sparse grid.

The default value of 'ZDS' is 1.

4.9.135 'ZDU' - Z Data Value, Unscaled - Real

The parameter 'ZDU' is just like 'ZDV' (which see, below), but the value is unscaled.

4.9.136 'ZDV' - Z Data Value - Real

The parameter 'ZDV' is mostly for output. Its value may be retrieved in a user version of CPCHHL to retrieve the value of the high or low which is being labelled. If a character string representing the value is desired, CPGETC may be used to obtain it (as modified by the current scale factor); thus, to obtain the character representation of an arbitrary value in a form consistent with the other values on a contour plot, set 'ZDV' with a call to CPSETR and retrieve the value of 'ZDV' with a call to CPGETC; if an unscaled value is desired, use the parameter name 'ZDU' in the call to CPGETC.

4.9.137 'ZMN' - Z Minimum Value - Real

The minimum value in the field, as found by CPRECT, CPSPS1, or CPSPS2. For output only.

4.9.138 'ZMX' - Z Maximum Value - Real

The maximum value in the field, as found by CPRECT, CPSPS1, or CPSPS2. For output only.

5.0 ERROR MESSAGES

When a CONPACK 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     3 IN CPGRWS - REAL WORKSPACE OVERFLOW
```

The error number (“3”, 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 (“CPGRWS”, 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 CONPACK 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.

When SETER is in recovery mode (and, occasionally, even when it is not), error messages may have a somewhat more complicated form, like this:

```
CPCLDR/CPGRWS - REAL WORKSPACE OVERFLOW
```

What this particular error message says is that CPCLDR called CPGRWS, which detected an error condition (real workspace overflow) and called SETER. Upon getting control back from CPGRWS, CPCLDR detected the fact that CPGRWS had logged an error. It augmented the error message by prepending its own name, followed by a slash, and then passed control back to the user. Of course, there can be more than two such levels of routine calls indicated in the error message: in a few cases, seven or eight routine names may be listed, each separated from the next by a slash.

The various error conditions in CONPACK are described in the list below. Each bulleted item includes an error message and a thumb-nail description of the error. The items in the list are arranged in alphabetical order. If you get an error message with one or more prefixed subroutine names, as described above, omit them and look up the result in this list. Note that, since CONPACK routines sometimes call other routines, elsewhere in NCAR Graphics, that can detect error conditions and call SETER, the error message you get by calling a CONPACK routine may not be listed here, but in the programmer document for some other package.

- **CPBACK - INITIALIZATION CALL NOT DONE**

  A call to CPRECT, CPSPS1, or CPSPS2 has been omitted.

- **CPBACK - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPBACK was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPCFLB - ERROR EXIT FROM GQCLIP**

An attempt to get the current clipping state has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPCFLB - ERROR EXIT FROM GQCNTN**

  An attempt to get the current normalization transformation number has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPCFLB - ERROR EXIT FROM GQFACI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPCFLB - ERROR EXIT FROM GQLWSC**

  An attempt to get a line width scale factor has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPCFLB - ERROR EXIT FROM GQNT**

  An attempt to get the current normalization transformation has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPCFLB - ERROR EXIT FROM GQPLCI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPCFLB - ERROR EXIT FROM GQTXCI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPCICA - CANNOT CONTINUE - CPMPXY DOES NOT DO INVERSE MAPPINGS**

  To use the routine CPCICA, it is necessary that the routine CPMPXY be able to do inverse mappings. The user has supplied his/her own version of CPMPXY, but has not made it capable of doing the inverses.

- **CPCICA - INITIALIZATION CALL NOT DONE**

  A call to CPRECT, CPSPS1, or CPSPS2 has been omitted.

- **CPCICA - ONE OF THE CORNER POINTS OF THE CELL ARRAY IS INCORRECT**

  One of the points P and Q, which determine how the cell array is to be mapped onto the plotter frame, is outside the limits of the plotter frame.

- **CPCICA - THE DIMENSIONS OF THE CELL ARRAY ARE INCORRECT**

  Either the specified first dimension of the FORTRAN array in which the cell array is stored is less than or equal to zero or one of the specified dimensions of the cell array itself is less than or equal to zero or the specified first dimension of the cell array is larger than the specified first dimension of the FORTRAN array in which it is stored.

- **CPCICA - UNCLEARED PRIOR ERROR**

This error message indicates that, at the time CPCICA was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPCLAM - CONTRADICTORY AREA-IDENTIFIER INFORMATION**

  The contour-level list has more than one entry for a given level (which is fine) but either the values of 'AIA' for that level or the values of 'AIB' for that level are inconsistent.

- **CPCLAM - INITIALIZATION CALL NOT DONE**

  A call to CPRECT, CPSPS1, or CPSPS2 has been omitted.

- **CPCLAM - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPCLAM was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPCLDM - ERROR EXIT FROM GQLWSC**

  An attempt to get a line width scale factor has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPCLDM - ERROR EXIT FROM GQPLCI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPCLDM - ERROR EXIT FROM GQTXCI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPCLDM - INITIALIZATION CALL NOT DONE**

  A call to CPRECT, CPSPS1, or CPSPS2 has been omitted.

- **CPCLDM - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPCLDM was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPCLDR - ERROR EXIT FROM GQLWSC**

  An attempt to get a line width scale factor has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPCLDR - ERROR EXIT FROM GQPLCI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPCLDR - ERROR EXIT FROM GQTXCI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPCLDR - INITIALIZATION CALL NOT DONE**

A call to CPRECT, CPSPS1, or CPSPS2 has been omitted.

- **CPCLDR - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPCLDR was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPCLTR - INITIALIZATION CALL NOT DONE**

  A call to CPRECT, CPSPS1, or CPSPS2 has been omitted.

- **CPCLTR - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPCLTR was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPCNRC - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPCNRC was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPEZCT - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPEZCT was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPGETC - GETTING X - PAI INCORRECT**

  An attempt has been made to get an element of the parameter array named 'X' and the current value of 'PAI' (the “parameter array index”) is inappropriate for that parameter array.

- **CPGETC - PARAMETER NAME NOT KNOWN - X**

  The given parameter name ('X') is not one of the legal parameter names.

- **CPGETC - PARAMETER NAME TOO SHORT - X**

  The parameter name ('X') is less than three characters long.

- **CPGETC - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPGETC was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPGETI - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPGETI was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPGETR - GETTING X - PAI INCORRECT**

  An attempt has been made to get an element of the parameter array named 'X' and the current value of 'PAI' (the “parameter array index”) is inappropriate for that parameter array.

- **CPGETR - PARAMETER NAME NOT KNOWN - X**

  The given parameter name ('X') is not one of the legal parameter names.

- **CPGETR - PARAMETER NAME TOO SHORT - X**

  The parameter name ('X') is less than three characters long.

- **CPGETR - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPGETR was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPGIWS - ARGUMENT ERROR - SEE SPECIALIST**

  This probably indicates an error in the implementation of the package. See the CONPACK specialist.

- **CPGIWS - INTEGER WORKSPACE OVERFLOW**

  The parameter 'WSO' has a value indicating that integer workspace overflow should be treated as a fatal error, and such an overflow has occurred.

- **CPGIWS m WORDS REQUESTED n WORDS AVAILABLE**

  This is not an error message used in a call to SETER; it is printed directly from the routine CPGIWS when an integer workspace overflow occurs and the value of 'WSO' is 1. The values “m” and “n” indicate how much additional integer workspace would be needed to continue executing the current CONPACK routine and how much additional integer workspace is currently available. Note that supplying “m” words on a subsequent attempt will get you past this point, but will not ensure immunity from a subsequent failure.

- **CPGRWS - ARGUMENT ERROR - SEE SPECIALIST**

  This probably indicates an error in the implementation of the package. See the CONPACK specialist.

- **CPGRWS - REAL WORKSPACE OVERFLOW**

  The parameter 'WSO' has a value indicating that integer workspace overflow should be treated as a fatal error, and such an overflow has occurred.

- **CPGRWS m WORDS REQUESTED n WORDS AVAILABLE**

  This is not an error message used in a call to SETER; it is printed directly from the routine CPGIWS when a real workspace overflow occurs and the value of 'WSO' is 1. The values “m” and “n” indicate how much additional real workspace would be needed to continue executing the current CONPACK routine and how much additional real workspace is currently available. Note that supplying “m” words on a subsequent attempt will get you past this point, but will not ensure immunity from a subsequent failure.

- **CPHLLB - ERROR EXIT FROM GQCNTN**

  An attempt to get the current normalization transformation number has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPHLLB - ERROR EXIT FROM GQFACI**

An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPHLLB - ERROR EXIT FROM GQLWSC**

  An attempt to get a line width scale factor has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPHLLB - ERROR EXIT FROM GQNT**

  An attempt to get the current normalization transformation has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPHLLB - ERROR EXIT FROM GQPLCI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPHLLB - ERROR EXIT FROM GQTXCI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPINLB - ERROR EXIT FROM GQCLIP**

  An attempt to get the current clipping state has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPINLB - ERROR EXIT FROM GQCNTN**

  An attempt to get the current normalization transformation number has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPINLB - ERROR EXIT FROM GQFACI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPINLB - ERROR EXIT FROM GQLWSC**

  An attempt to get a line width scale factor has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPINLB - ERROR EXIT FROM GQNT**

  An attempt to get the current normalization transformation has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPINLB - ERROR EXIT FROM GQPLCI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPINLB - ERROR EXIT FROM GQTXCI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPLBAM - INITIALIZATION CALL NOT DONE**

  A call to CPRECT, CPSPS1, or CPSPS2 has been omitted.

- **CPLBAM - UNCLEARED PRIOR ERROR**

This error message indicates that, at the time CPLBAM was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPLBDR - ERROR EXIT FROM GQCLIP**

  An attempt to get the current clipping state has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPLBDR - ERROR EXIT FROM GQFACI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPLBDR - ERROR EXIT FROM GQLWSC**

  An attempt to get a line width scale factor has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPLBDR - ERROR EXIT FROM GQPLCI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPLBDR - ERROR EXIT FROM GQTXCI**

  An attempt to get a color index has resulted in an error. This probably indicates that GKS is in the wrong state.

- **CPLBDR - INITIALIZATION CALL NOT DONE**

  A call to CPRECT, CPSPS1, or CPSPS2 has been omitted.

- **CPLBDR - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPLBDR was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPMVIW - NEW WORKSPACE ARRAY IS TOO SMALL**

  The new integer workspace array is too small to contain what was in the old one.

- **CPMVIW - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPMVIW was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPMVRW - NEW WORKSPACE ARRAY IS TOO SMALL**

  The new real workspace array is too small to contain what was in the old one.

- **CPMVRW - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPMVRW was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPPKCL - INITIALIZATION CALL NOT DONE**

  A call to CPRECT, CPSPS1, or CPSPS2 has been omitted.

- **CPPKCL - TOO MANY CONTOUR LEVELS**

  Indicates that the parameter 'CLS' has been given a negative value whose absolute value is too large, requesting more than 256 contour levels.

- **CPPKCL - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPPKCL was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPPKLB - INITIALIZATION CALL NOT DONE**

  A call to CPRECT, CPSPS1, or CPSPS2 has been omitted.

- **CPPKLB - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPPKLB was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPPKLP - INITIALIZATION CALL NOT DONE**

  A call to CPRECT, CPSPS1, or CPSPS2 has been omitted.

- **CPPKLP - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPPKLP was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPRECT - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPRECT was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPRSET - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPRSET was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPSETC - PARAMETER NAME NOT KNOWN - X**

  The given parameter name ('X') is not one of the legal parameter names.

- **CPSETC - PARAMETER NAME TOO SHORT - X**

  The parameter name ('X') is less than three characters long.

- **CPSETC - SETTING X - PAI INCORRECT**

  An attempt has been made to set an element of the parameter array named 'X' and the current value of 'PAI' (the “parameter array index”) is inappropriate for that parameter array.

- **CPSETC - UNCLEARED PRIOR ERROR**

This error message indicates that, at the time CPSETC was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPSETI - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPSETI was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPSETR - NCL LESS THAN 1 OR GREATER THAN n**

  An attempt has been made to set the number of contour levels to an illegal value. The value of “n” is the largest value which may be used.

- **CPSETR - PARAMETER NAME NOT KNOWN - X**

  The given parameter name ('X') is not one of the legal parameter names.

- **CPSETR - PARAMETER NAME TOO SHORT - X**

  The parameter name ('X') is less than three characters long.

- **CPSETR - SETTING X - PAI INCORRECT**

  An attempt has been made to set an element of the parameter array named 'X' and the current value of 'PAI' (the “parameter array index”) is inappropriate for that parameter array.

- **CPSETR - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPSETR was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPSPRS - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPSPRS was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPSPS1 - CANNOT CONTINUE WITHOUT WORKSPACE**

  Insufficient workspace has been provided for the execution of CPSPS1. The amount of space which it requires is entirely predictable. See the section entitled “Workspace Management”, above.

- **CPSPS1 - IZD1, IZDM, OR IZDN SET INCORRECTLY**

  IZDS has been given a zero value, indicating that the user intends to supply the values of IZD1, IZDM, and IZDN, and their current values are incorrect.

- **CPSPS1 - SPECIAL-VALUE REPLACEMENT FAILURE**

  There are no two adjacent values in the sparse array which are not special values. Most likely, the entire sparse array is filled with special values.

- **CPSPS1 - UNCLEARED PRIOR ERROR**

This error message indicates that, at the time CPSPS1 was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPSPS2 - CANNOT CONTINUE WITHOUT WORKSPACE**

  Insufficient workspace has been provided for the execution of CPSPS2. The amount of space which it requires is entirely predictable. See the section entitled "Workspace Management", above.

- **CPSPS2 - ERROR IN CALL TO MSSRF1**

  A non-zero error return has occurred from the FITPACK routine SURF1. This probably means that the X and Y coordinate arrays are not in strictly increasing numerical order.

- **CPSPS2 - IZD1, IZDM, OR IZDN SET INCORRECTLY**

  IZDS has been given a zero value, indicating that the user intends to supply the values of IZD1, IZDM, and IZDN, and their current values are incorrect.

- **CPSPS2 - SPECIAL-VALUE REPLACEMENT FAILURE**

  There are no two adjacent values in the sparse array which are not special values. Most likely, the entire sparse array is filled with special values.

- **CPSPS2 - UNCLEARED PRIOR ERROR**

  This error message indicates that, at the time CPSPS2 was called, there was an unrecovered outstanding error. In this case, it cannot continue; it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **CPTRES - ALGORITHM FAILURE - SEE SPECIALIST**

  An error has occurred while trying to trace the edge of a special-value area. This should not be possible. See the CONPACK specialist.

6.0 EXAMPLES

Thirteen examples are available for CONPACK. On a Unix system on which NCAR Graphics has been installed, one may obtain the code for an example and run it by executing the command

```
ncargex example_name
```

where "example_name" is one of the following:

```
tconpa cpex01 cpex02 cpex03 cpex04 cpex05 cpex06 cpex07
cpex08 cpex09 cpex10 cpex11 cpex12 cpex13
```

To obtain the code for an example without running it, one uses the command

```
ncargex -n example_name
```

After the code has been examined and, perhaps, modified, it can be compiled and executed using the commands

```
ncargf77 *.f
a.out
```

The metafile created by executing an example can be viewed using the command

```
ctrans metafile_name
```

(It may be necessary to set some environment variables and/or to set other options on the "ctrans" command line.)

See the document "Using NCAR Graphics in a Unix Environment" for additional details.

6.1 The Example "tconpa"

The example "tconpa" produces three frames, as follows:

- The first frame demonstrates the use of the routine CPEZCT to produce a quick-and-dirty contour plot of a simple set of data.
- The second frame demonstrates the use of the routine CPCNRC to produce a slightly more polished contour plot of the same set of data.
- The third frame demonstrates the use of the underlying CONPACK routines to produce a color-filled contour plot of the same set of data.

This example is intended to demonstrate minimal functioning of the package, rather than to illustrate the ins and outs of how to use CONPACK.

6.2 The Examples "cpex01" through "cpex09"

The example "**cpex01**" produces six frames, identified as "1-1", "1-2", etc., illustrating a progression from a very simple contour plot to one of considerable complexity. The same array of data is used for all six.

- Example 1-1 was produced as efficiently as possible, using the default settings of CONPACK parameters and the lowest-quality characters.
- Example 1-2 was drawn using various features to produce a more attractive plot. Characters of medium quality were used.
- Example 1-3 is the same as example 1-2, but with shading of that part of the plot where the data values are less than zero.
- Example 1-4 is the same as example 1-3, but with the 2D smoother turned on. Note that there are two places on the plot where the 2D smoothing has caused contour lines to touch one another. This can be corrected by using a greater value for 'T2D', the tension on the splines. (In this particular case, the value 4.0 works well.)
- Example 1-5 uses CPSPRS to do 3D interpolation. The resulting curves are very smooth. The range of the data has been increased somewhat. Note that the size of ZDNS (5152) is exactly 16 times the size of ZDAT (23 x 14); as a result, CPSPRS uses 'ZDM' = 92 (4 x 23) and 'ZDN' = 56 (4 x 14), thus preserving the same aspect ratio for

the data grid. If the size of ZDNS had not been an exact multiple of the size of ZDAT and a perfect square, the dimensions might have been chosen in such a way as to give a little different aspect ratio. This may or may not be of importance to the user.

- Example 1-6 shows the same plot as that of example 1-5, but it is mapped onto a satellite-view projection of the globe.

The example “**cpex02**” demonstrates the capability of putting more than one contour plot on a frame. It also illustrates how to shade the area between two different contour levels and shows four of the ways in which a scale factor may be selected.

The example “**cpex03**” demonstrates the use of various kinds of mappings and the use of a special value to signal missing data in a portion of the input data array. The default version of CPMPXY is replaced by an alternate version which does the same mapping as the default for 'MAP' = 1 or 2, but does two other useful mappings as well. In example 3-1, the contour plot is mapped onto an EZMAP background; longitude is a linear function of the first subscript and latitude a linear function of the second subscript. In example 3-2, the plot is mapped onto a polar-coordinate background; rho is a linear function of the first subscript and theta a linear function of the second subscript. In example 3-3, the grid is rectangular, but the X and Y coordinates are not regularly spaced; their positions are specified by singly-dimensioned arrays in a labelled common block. In example 3-4, the grid is completely distorted into the shape of a four-pointed star; the X and Y coordinates corresponding to each index pair (I,J) are specified by doubly-dimensioned arrays provided in another labelled common block.

The example “**cpex04**” demonstrates how to specify all contour levels and associated information exactly. Intermediate contour levels, for which only labels, but not lines, are drawn, are used to effect a labelling of the contour bands defined by the other levels; the results are adequate, if not spectacular. The package SOFTFILL is used for shading.

The example “**cpex05**” shows the same data as example 4. Twenty levels of shading are used. No labels are placed on the plot; instead, a label bar is used to indicate the values associated with the shading.

The example “**cpex06**” shows the use of color, but not for solid fill. Contour lines for positive values are shown in red, those for negative values are shown in blue, and the zero contour is shown in white. Highs and lows are labelled with just the number (no 'H' or 'L') and are in green. The range of values is intentionally such as to force the use of exponents in all numeric labels; a scale factor could, of course, be used in order to avoid this.

The example “**cpex07**” shows the same data as example 1, but with color fill of contour bands and with a color-filled label bar.

The example “**cpex08**” shows a color-filled contour plot on the surface of the earth, in orthographic projection, with a color bar to show what the colors mean. It is assumed that the field being contoured has meaning only over the African continent and, accordingly, contours are only shown there. Lines of latitude and longitude are drawn only over the ocean areas.

The example “**cpex09**” shows four plots drawn by the routines CPEZCT and CPCNRC.

A number of routines are used in more than one of the above examples; these routines are all in a file “cpexcc.f” that “ncargex” copies to your directory. Many of them do useful things, are of interest to users, and may eventually, in some form, become officially supported utility routines in their own right. Thumbnail descriptions are as follows:

- The routine **GENDAT** is called to generate dummy data.
- The function **FRAN** is called by GENDAT; it simulates the behavior of a random number generator, but produces a predictable string of numbers, so that the exact plots shown in the examples may be duplicated.
- The routine **CAPSAP** is called to get and print statistics about each plot or subplot.
- The routine **LABTOP** is called to put a label above the current viewport.
- The routine **BNDARY** is called to draw the boundary of the plotter frame.
- The routine **DFCLRS** is called to define a set of color indices.
- The routine **SECOND** is used to get the elapsed CPU time, in seconds. The version provided is a dummy that always returns the value 0. If you really want to time the examples, you should replace this routine with one of your own.

6.3 The Example “cpex10”

The example “**cpex10**” produces a single frame illustrating the use of the satellite-view projection and various routines from the package EZMAPA. It shows a view of the western United States, as seen from a satellite above Washington, D.C. In the area within seven degrees of Boulder, Colorado, color-filled contour bands are shown. The EZMAPA routine MAPBLM is used with an appropriate area map, allowing state boundaries to be shown in black over the contoured area and in white elsewhere. The EZMAP routine MAPGRM is used with an appropriate area map, allowing 1-degree grid lines to be drawn outside the contoured area, but not inside it. PLOTCHAR is used with 'MA' = 4 to create state labels that appear to be written on the surface of the globe and projected with it.

6.4 The Example “cpex11”

The example “**cpex11**” produces five frames illustrating the sort of problem that the internal parameters 'PIC', 'PIE', and 'PIT' were intended to address. The first frame shows a Lambert equal-area projecton of a contour map on the surface of the globe; distortion near the limb of this projection causes some contour lines to cross each other. The second frame shows what happens when 'PIC' and 'PIE' are given large enough values: contour lines no longer cross each other, but the size of the metafile is seriously increased and another kind of distortion occurs in areas where points map very close together (use the “zoom” capability of “idt” to take a close look at the wiggly chunk of contour line on the extreme right edge of the frame). The third frame shows what happens when 'PIT' is given the value .05; contour lines no longer cross, the metafile is not much larger than when 'PIT' is zero, and no other types of distortion show up. The fourth and fifth frames simply show that smoothing can be done when 'PIT' is non-zero; this may be of more interest to the author than to the reader (it considerably complicated the code required to implement 'PIT').

6.5 The Example “cpex12”

The example “**cpex12**” produces a single frame illustrating the use of the new dashed-line package DASHPACK by CONPACK.

6.6 The Example “cpex13”

The example “**cpex13**” produces four frames showing what the CONPACK routines CPCLAM and CPLBAM put into an area map and some situations that can cause trouble. The frames are drawn using a slightly-modified version of the AREAS debugging routine ARDBPA; the latter is described in the programmer document for AREAS and is recommended as an aid in diagnosing problems with area maps created by CONPACK.

Each frame produced by “cpex13” shows the contents of edge group 3 in an area map produced by calling the AREAS routine ARINAM and the CONPACK routines CPCLAM and (for the first two frames only) CPLBAM. By default, edge group 3 contains the following (in the order in which they are put into the area map):

- the edge of the plotter frame (actually provided by the AREAS routine AREDAM);
- the edge of the viewport;
- edges separating areas visible under the mapping from areas invisible under the mapping (applicable only if mapping is activated by setting 'MAP' non-zero and an “out-of-range” value is defined by setting 'ORV' non-zero);
- the edge of the grid;
- the edge of the special-value area (applicable only if a special value is defined by setting 'SPV' non-zero and this value actually occurs in the array of data being contoured);
- contour lines (applicable only if ‘NCL’ is non-zero and one or more elements in the parameter arrays 'AIA' and 'AIB' are non-zero);
- label boxes (applicable only if ABS['LLP'] is given the value 2 or 3 and CPLBAM is called).

Each edge segment shown is accompanied by its two area identifiers, each of which is written in the appropriate position relative to the midpoint of the edge segment. Labels point out features of interest.

6.6.1 Frame 1

Frame 1 just shows a typical situation.

The **edge of the plotter frame** has area identifiers that are 0 on the inside and -1 on the outside; this is not under user control, but is dictated by the AREAS routine AREDAM.

The **edge of the viewport** has area identifiers that are 0 on the inside and -1 on the outside; this is not under user control, but is dictated by the CONPACK routine ARCLAM. In this case, the edge of the viewport is outside the edge of the grid; usually, it is exactly on top of the grid, and, sometimes, it is inside the grid.

The **edge of the invisible area** has area identifiers that are -1 on the invisible side and variable on the visible side. The value on the invisible side is user-settable as element “-3” in the internal parameter array 'AIA'; use the code

```
CALL CPSETI ('PAI - PARAMETER ARRAY INDEX',-3)
CALL CPSETI ('AIA - AREA IDENTIFIER ARRAY',n)
```

to reset the value to “n”.

In general, it is very difficult to generate this edge in pieces such that each piece is associated with exactly one contour band. The area identifier on the visible side of each piece is non-zero only if the piece does seem to be associated with just one contour band, and zero otherwise.

If you use the “zoom” feature of “idt” to look closely at this edge, you will see that it is generated twice, once slightly offset to the left and again slightly offset to the right. This is done to ensure that a contour line dead-ending at the edge of visibility will intersect at least one of the two. If it were not done, AREAS might decide to edit the contour line out. (On frame 2, we will see a different sort of visible/invisible edge, generated when the mapping function used does not do inverses.)

The **edge of the grid** has area identifiers that are 0 on the outside and variable on the inside. The value on the outside is user-settable as element “-1” in the internal parameter array 'AIA'; use the code

```
CALL CPSETI ('PAI - PARAMETER ARRAY INDEX',-1)
CALL CPSETI ('AIA - AREA IDENTIFIER ARRAY',n)
```

to reset the value to “n”. The grid edge is generated in pieces in such a way that each piece is entirely within a given contour band (or a special-value area), so that its inside area identifier can be unambiguously determined.

The **edge of the special-value area** has area identifiers that are -1 on the special-value side and variable on the other side. The value on the special-value side is user-settable as element “-2” in the internal parameter array 'AIA'; use the code

```
CALL CPSETI ('PAI - PARAMETER ARRAY INDEX',-2)
CALL CPSETI ('AIA - AREA IDENTIFIER ARRAY',n)
```

to reset the value to “n”. This edge is generated in pieces in such a way that each piece is entirely within a given contour band, so that its non-special-value side can be unambiguously determined.

Each of the four **contour lines** has an area identifier associated with its “downhill” side, where values are less than the value defining the contour line, and a different area identifier associated with its “uphill” side, where values are greater than the value defining the contour line. These values are defined by entries in the internal parameter arrays 'AIA' and 'AIB'.

Note that the value on the “uphill” side of each contour is made consistent with the value on the “downhill” side of the next higher contour. If this is not done, problems result.

Each of the **label boxes** is defined by four edge segments, with area identifiers that are zero on the outside and -1 on the inside. These values are not user-settable, but are dictated by the routine CPLBAM.

It is the job of **AREAS** to digest all of this information, determine what areas are defined by the edges in the area map, and reconcile conflicting area-identifier information in such a way as to come up with the desired area identifier for each of those areas. For more information about AREAS, see the programmer document for that package. Also see the AREAS examples “arex01” and “arex03”.

6.6.2 Frame 2

Frame 2 is much like frame 1, but the mapping function used doesn’t know how to do inverses (or, at least, it says that it doesn’t). In this case, the edge of the invisible area is generated in a rather awkward way: First, we determine a small set of points known to lie on the exact edge of visibility (those points where that edge crosses the edges of data-grid cells). Then, if the edge is determined to be convex as seen from the visible side, other points are interpolated in such a way as to try to ensure that the polygonal approximation generated lies entirely within the visible area. The algorithm is error-prone, and, if the edge of visibility is S-shaped (partly convex and partly concave as viewed from the visible side), the algorithm is almost sure to fail. When it fails, the result will be that a contour line dead-ending at the edge of visibility goes into the area map with a gap between it and the polyline representing the edge of visibility; AREAS sees this as a “dangling end” and edits it out, probably causing two adjacent contour bands to be merged into one.

Even when the algorithm doesn’t fail, if the data grid is relatively coarse (as is the case here), the generated boundary of the invisible area will be rather rough. If you really can’t make the mapping routine do inverses, you can use the internal parameter 'PIE' to improve the situation somewhat. Setting ‘PIE' to “n” has the effect of making the algorithm behave as if each edge of each cell of the data grid were subdivided into n+1 equal pieces; this causes each cell to be subdivided into $(n+1)^2$ subcells, which causes more points exactly on the edge of visibility to be generated and improves the approximation to the actual visible/invisible edge. Unfortunately, as “n” gets bigger, this becomes very time-consuming.

Only one copy of the edge is generated (rather than the two offset copies that are generated when inverses are available).

6.6.3 Frame 3

Frame 3 illustrates a problem that occurs when the values in the data field are exactly as requred to create a contour-line segment that, to the accuracy of the area map (which uses integer coordinates between 1 and 1,000,000 to represent the plotter frame), appears to lie exactly on top of the edge of the grid. In this case, the contour-line segment and a congruent segment of the edge of the grid end up being treated by AREAS as “coincident segments”. One of them is deleted from the area map; the other is left in, but its area identifiers are changed. In versions of NCAR Graphics prior to Version 4, this worked in such a way as to sometimes leave a non-zero area identifier on the ouside of the grid. In effect, the color for a contour band could “leak” outside the grid. This has been fixed in version 4; it remains to be seen if the fix causes problems in other situations.

6.6.4 Frame 4

Frame 4 illustrates a problem that occurs when the mapping function in use transforms two portions of the data grid into the same line. This is similar to the problem described for frame 3, and the fix that went into version 4 for that problem should also apply to this one. This problem becomes more serious when two portions of the data grid map into part of a curved line, rather than part of a straight line (as is frequently the case when using an EZMAP transformation). Also, if you change the area identifier to be used outside the grid from the default “0” to “-1”, it may affect the situation. Code has been put in to avoid such problems, but there may still be cases in which problems arise.