DASHPACK, A SOFTWARE PACKAGE FOR DRAWING DASHED LINES

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

DASHPACK, A SOFTWARE PACKAGE FOR DRAWING DASHED LINES

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

1.0 INTRODUCTION

The NCAR Graphics package DASHPACK draws curves using dashed-line patterns that may include gap-portion specifiers, solid-portion specifiers, and label-string specifiers. DASHPACK is intended to replace and unify the old family of NCAR Graphics routines (DASHLINE, DASHCHAR, DASHSMTH, and DASHSUPR), all of which date from the early 1970s or before. At the time of its introduction, DASHPACK will do anything that DASHLINE, DASHCHAR, and DASHSMTH will (and a number of things that they won't); it does not yet functionally replace all of DASHSUPR, but it is intended that this will eventually happen. For the moment, the older routines will stay in NCAR Graphics.

1.1 Differences between DASHPACK and the Routines it Replaces

Structurally, DASHPACK is rather different from the routines that it replaces: Its behavior is controlled by "internal parameters" that are accessed by calling parameter-access routines, rather than by manipulating the contents of labelled common blocks. There is only one version of DASHPACK; rather than load a different version of it in order to, say, turn smoothing on, one just changes the value of the internal parameter that controls whether or not smoothing is done. There is no analog of the old routines DASHDB and DASHDC in DASHPACK; to use a different dash pattern, one just changes the values of the internal parameters that specify what dash pattern to use.

1.2 Internal Parameters

As in other NCAR Graphics packages, each of the internal parameters of DASHPACK has a three-character mnemonic name. The internal parameters are described in detail in the first-level section called "INTERNAL PARAMETERS"; in the rest of this section, as a feature is mentioned, the names of the pertinent internal parameters are given, in parentheses, for reference.

1.3 Dashed-Line Patterns

A dashed-line pattern (or just “dash pattern”) may be specified using either an integer or a character string (see the parameters 'DPL', 'DPS', and 'DPT'). However the pattern is specified, it is applied in a repetitive fashion as a curve is drawn. For example, if the dash pattern specifies a solid, a gap, and the label “A”, a curve drawn using it will include a solid, a gap, the label “A”, a solid, a gap, the label “A”, and so on, until the end of the curve is reached. When a new curve is begun, use of the dash pattern starts over: the new curve starts with a solid, a gap, and the label “A”.

If an integer dash pattern is used, the low-order “n” bits of the integer (where “n” is the absolute value of the user-specified parameter, 'DPS', and is between 1 and 32, inclusive) are interpreted; 0’s represent gaps and 1’s represent solids.

In a character-string dash pattern, the character used to represent a gap (by default, an underscore) and the character used to represent a solid (by default, a dollar sign) are determined by the values of other internal parameters ('CRG' and 'CRS'); such characters may be mixed with other strings of characters that are to be used as label strings. There are parameters allowing one to specify how much distance along the curve should be devoted to each gap ('WOG') and solid ('WOS'), what size characters should be used for writing label strings ('WOC'), whether label strings should be written at a fixed angle relative to the plotter frame or in the direction of the curve ('SAF'), and, if they are written in the direction of the curve, whether or not they should be made to “bend” with the curve ('CRB' and 'SCF') and whether or not it is allowed to rotate them by 180 degrees to make them more nearly upright on the plotter frame ('SAF').

A label string written along a curve may either be placed in a gap left in the curve or just written on top of it ('LS1', 'LS2', and 'LTL'); the latter is most effective when the characters are written in a color different from that of the curve itself.

Output of label strings is buffered so that, if the end of the curve occurs prior to the occurrence of the next solid or gap in the dash pattern, the curve itself may be drawn in lieu of an incomplete label, and so that, when labels are being made to “bend” with the curve, the entire label can be inverted, if necessary, so as to have the majority of it right-side up. This buffering is done by default, but may be turned off, if desired ('SBF').

All character strings are written using calls to PLOTCHAR routines. There is an internal parameter ('PCF') that says whether to call PLCHHQ, PLCHMQ, or PLCHLQ. By default, PLCHHQ is used.

1.4 Curve Smoothing

Curve smoothing is turned on and off by setting a single internal parameter ('TCS') that also specifies the tension on the cubic splines used to do the smoothing. Another parameter ('SSL') specifies the distance between points used to plot the smoothed curve and another ('EPS') specifies the minimum distance between input points required for them to be considered separate points.

1.5 Curve-Drawing Routines That Accept User Coordinates

The following routines are called with arguments specifying the user coordinates (in the ranges specified by arguments 5 through 8 of a preceding call to SET) of points defining a curve to be drawn using the current dash pattern. With the exception of DPLINE, these routines react to the internal parameter that requests smoothing:

- **DPCURV** draws a complete curve with a single call.
- **DPLINE** draws the straight line joining two points. (No smoothing takes place, even if smoothing is turned on.)
- **DPFRST** does a "pen-up" move to the first in a sequence of points defining a curve.
- **DPVECT** does a "pen-down" move to the next in a sequence of points defining a curve.
- **DPLAST** is called to say that the latest call to DPVECT was the last one in the sequence, that drawing of the curve should be completed, and that buffers should be flushed.

Each of the above routines just converts the incoming coordinates from the user coordinate system to the fractional coordinate system (in GKS parlance, NDC's) and then passes those on to one of the routines DPDRAW or DPSMTH.

1.6 Curve-Drawing Routines That Accept Fractional Coordinates

The following routines are called with arguments specifying the fractional coordinates (from 0 to 1) of points defining a curve to be drawn with the current dash pattern:

- **DPDRAW** is called to just draw the curve by connecting a given sequence of points.
- **DPSMTH** is called to create a smooth curve passing through a sequence of points and then to interpolate points along that smoothed curve and pass them on to DPDRAW.

1.7 Parameter-Setting Routines

The parameter-setting routines are consistent with those in other NCAR Graphics utilities. Each of the routines DPSETC, DPSETI, and DPSETR is intended to give new values to a particular subset of the parameters of DASHPACK, as follows:

- **DPSETC** is called with the name of an internal parameter of type CHARACTER and a character variable or constant that is to supply the new value of that parameter.
- **DPSETI** is called with the name of an internal parameter of type INTEGER or REAL and an integer expression that is to become the new value of that parameter.
- **DPSETR** is called with the name of an internal parameter of type INTEGER or REAL and a real expression that is to become the new value of that parameter.

If, at the time any one of the parameter-setting routines is called, the internal buffers of DPDRAW or DPSMTH are non-empty, those buffers are flushed before the parameter is changed. This ensures that parameter changes will affect only curves that are defined by curve-drawing calls following the parameter-change call.

1.8 Parameter-Retrieval Routines

The parameter-retrieval routines are consistent with those in other NCAR Graphics utilities. Each of the routines DPGETC, DPGETI, and DPGETR is intended to retrieve the values of a particular subset of the internal parameters of DASHPACK:

- **DPGETC** is called with the name of an internal parameter of type CHARACTER and a character variable in which the value of that parameter is to be returned.
- **DPGETI** is called with the name of an internal parameter of type INTEGER or REAL and an integer variable in which the integer equivalent of the value of that parameter is to be returned.
- **DPGETR** is called with the name of an internal parameter of type INTEGER or REAL and a real variable in which the real equivalent of the value of that parameter is to be returned.

2.0 ROUTINES

2.1 DPCURV (XCPU,YCPU,NPTS)

Draws a complete curve defined by a sequence of points in the user coordinate system.

2.1.1 Usage

The Fortran statement

```
CALL DPCURV (XCPU,YCPU,NPTS)
```

draws a complete curve by connecting the point (XCPU(1),YCPU(1)) to the point (XCPU(2),YCPU(2)), connecting that point to the point (XCPU(3),YCPU(3)), and so on through the last point, (XCPU(NPTS),YCPU(NPTS)), in a manner described by the current values of the internal parameters of DASHPACK.

DPCURV maps the input coordinates to the fractional system and then passes them on to lower-level routines. If smoothing is turned off, DPCURV calls DPDRAW; if smoothing is turned on, it calls DPSMTH.

2.1.2 Arguments

XCPU and **YCPU** (input arrays of type REAL, dimensioned NPTS) contain the X and Y coordinates of NPTS points defining the curve that is to be drawn.

NPTS (an input expression of type INTEGER) is the number of points defining the curve.

2.2 DPLINE (XCP1,YCP1,XCP2,YCP2)

Draws a straight-line segment between two points in the user coordinate system.

2.2.1 Usage

The Fortran statement

```
CALL DPLINE (XCP1,YCP1,XCP2,YCP2)
```

draws a straight-line segment joining the point (XCP1,YCP1) to the point (XCP2,YCP2) in a manner described by the current values of the internal parameters of DASHPACK.

DPLINE maps the input coordinates to the fractional system and then passes them on to DPDRAW. Even if smoothing is turned on, no smoothing is done by a call to DPLINE.

2.2.2 Arguments

XCP1 and **YCP1** (input expressions of type REAL) are the X and Y coordinates, in the user coordinate system, of the point at the beginning of the straight-line segment.

XCP2 and **YCP2** (input expressions of type REAL) are the X and Y coordinates, in the user coordinate system, of the point at the end of the straight-line segment.

2.3 DPFRST (XCPU,YCPU)

Specifies the first point in a sequence of points, in the user coordinate system, defining a curve to be drawn.

2.3.1 Usage

The Fortran statement

```
CALL DPFRST (XCPU,YCPU)
```

does a “pen-up” move to the first of a sequence of points defining a curve. Call DPVECT to do “pen-down” moves to each of the other points in the sequence and then call DPLAST to finish drawing the curve and flush the buffers.

DPFRST maps the input coordinates of the given point to the fractional coordinate system and then does a “pen-up” call to either DPDRAW (if smoothing is turned off) or DPSMTH (if smoothing is turned on) with those coordinates.

2.3.2 Arguments

XCPU and **YCPU** (input expressions of type REAL) are the X and Y coordinates of a point in the user coordinate system.

2.4 DPVECT (XCPU,YCPU)

Specifies the second and following points in a sequence of points, in the user coordinate system, defining a curve to be drawn.

2.4.1 Usage

The Fortran statement

```
CALL DPVECT (XCPU,YCPU)
```

does a “pen-down” move to each point after the first in a sequence of points defining a curve. Prior to the first call to DPVECT, call DPFRST to do a “pen-up” move to the first point in the sequence and then, after the last call to DPVECT, call DPLAST to finish drawing the curve and flush the buffers.

DPVECT maps the input coordinates of the given point to the fractional coordinate system and then does a “pen-down” call to either DPDRAW (if smoothing is turned off) or DPSMTH (if smoothing is turned on) with those coordinates.

2.4.2 Arguments

XCPU and **YCPU** (input expressions of type REAL) are the X and Y coordinates of a point in the user coordinate system.

2.5 DPLAST

Calling DPLAST says that the preceding call to DPVECT specified the final point in a sequence of points, in the user coordinate system, defining a curve to be drawn.

2.5.1 Usage

Use the Fortran statement

```
CALL DPLAST
```

to finish drawing the curve defined by prior calls to DPFRST and DPVECT and to flush the buffers. Call DPFRST to do a “pen-up” move to the first point in the sequence, DPVECT to do “pen-down” moves to the second and following points in the sequence, and DPLAST to finish the process.

DPLAST just does a “buffer-flush” call to either DPDRAW (if smoothing is turned off) or DPSMTH (if smoothing is turned on).

2.5.2 Arguments

None.

2.6 DPDRAW (XCPF,YCPF,IFVL)

Used to draw a curve when fractional coordinates are available and smoothing is **not** to be done.

2.6.1 Usage

Use the Fortran statement

```
CALL DPDRAW (XCPF,YCPF,0)
```

to do a “pen-up” move to the first point in a sequence of points defining a curve, Fortran statements like

```
CALL DPDRAW (XCPF,YCPF,1)
```

to do pen-down moves to the second and following points in the sequence, and the Fortran statement

```
CALL DPDRAW (0.,0.,2)
```

to terminate the sequence, finish drawing the curve, and flush the buffers.

2.6.2 Arguments

XCPF and **YCPF** (input expressions of type REAL), when not ignored, are the X and Y coordinates of a point in the fractional coordinate system.

IFVL (an input expression of type INTEGER) is a flag indicating what kind of call is being done:

- IFVL=0 implies a "first-point" call; (XCPF,YCPF) is the first point of a curve.
- IFVL=1 implies a "vector" call; (XCPF,YCPF) is a continuation point on the curve.
- IFVL=2 implies a "last-point" or "buffer-flush" call; XCPF and YCPF are ignored, but the curve being drawn is finished and all buffers are flushed.

Actually, DPDRAW is rather more flexible than the above may imply: If a "first-point" call is omitted, the first "vector" call will act like a "first-point" call. Similarly, if a "last-point" call is omitted, the next "first-point" call will finish off the last curve and flush the buffers before starting a new curve. Nevertheless, it is a good idea to draw each curve using one "first-point" call, one or more "vector" calls, and a final "last-point" call.

2.7 DPSMTH (XCPF,YCPF,IFVL)

Used to draw a curve when fractional coordinates are available and smoothing **is** to be done.

2.7.1 Usage

Use the Fortran statement

```
CALL DPSMTH (XCPF,YCPF,0)
```

to do a "pen-up" move to the first point in a sequence of points defining a curve, Fortran statements like

```
CALL DPSMTH (XCPF,YCPF,1)
```

to do pen-down moves to the second and following points in the sequence, and the Fortran statement

```
CALL DPSMTH (0.,0.,2)
```

to terminate the sequence, finish drawing the curve, and flush the buffers.

DPSMTH accumulates points until it has an internal buffer-load or until the last one has been received. It then creates a smooth curve passing through all the points, interpolates points along that smooth curve, and passes those points along to the routine DPDRAW. If the internal parameter 'TCS' has a value less than or equal to zero, the smoothing is done using simple cubic splines; if 'TCS' has a value greater than zero, the smoothing is done using cubic splines under tension and 'TCS' specifies the desired tension.

2.7.2 Arguments

XCPF and **YCPF** (input expressions of type REAL), when not ignored, are the X and Y coordinates of a point in the fractional coordinate system.

IFVL (an input expression of type INTEGER) is a flag indicating what kind of call is being done:

- IFVL=0 implies a “first-point” call; (XCPF,YCPF) is the first point of a curve.
- IFVL=1 implies a “vector” call; (XCPF,YCPF) is a continuation point on the curve.
- IFVL=2 implies a “last-point” or “buffer-flush” call; XCPF and YCPF are ignored, but the curve being drawn is finished and all buffers are flushed.

Actually, DPSMTH, like DPDRAW, is rather more flexible than the above may imply: If a “first-point” call is omitted, the first “vector” call will act like a “first-point” call. Similarly, if a “last-point” call is omitted, the next “first-point” call will finish off the last curve and flush the buffers before starting a new curve. Nevertheless, it is a good idea to draw each curve using one “first-point” call, one or more “vector” calls, and a final “last-point” call.

2.8 DPSETx (PNAM,xVAL)

Gives a new value to a specified internal parameter.

2.8.1 Usage

Use one of the statements

```
CALL DPSETC (PNAM,CVAL)
CALL DPSETI (PNAM,IVAL)
CALL DPSETR (PNAM,RVAL)
```

to set the value of the parameter specified by PNAM to the value specified by the argument CVAL, IVAL, or RVAL.

Before changing the parameter, DPSETx will do a

```
CALL DPSMTH (0.,0.,2)
```

to flush all buffers in DPSMTH and DPDRAW. This ensures that the parameter change will affect only curves drawn by later calls.

2.8.2 Arguments

PNAM (an input variable or constant of type CHARACTER) is a string of three or more characters, the first three characters of which constitute the name of the internal parameter whose value is to be set.

xVAL (an input variable or constant of type CHARACTER or an input expression of type INTEGER or REAL, depending on whether “x” is a “C”, an “I”, or an “R”) defines the new value to be given to the internal parameter specified by PNAM.

- If “x” is a “C”, the internal parameter must be of type CHARACTER.
- If “x” is an “I” or an “R”, the internal parameter may be of type INTEGER or REAL; if the type of “xVAL” and the type of the internal parameter do not match, the effect will be the same as that of a mixed-mode arithmetic replacement statement in FORTRAN.

For more information, see the section “INTERNAL PARAMETERS”.

2.9 DPGETx (PNAM,xVAL)

2.9.1 Usage

Use one of the Fortran statements

```
CALL DPGETC (PNAM,CVAL)
CALL DPGETI (PNAM,IVAL)
CALL DPGETR (PNAM,RVAL)
```

to retrieve, in the variable CVAL, IVAL, or RVAL, the value of the parameter specified by PNAM.

2.9.2 Arguments

PNAM (an input variable or constant of type CHARACTER) is a string of three or more characters, the first three of which constitute the name of the internal parameter whose value is to be retrieved.

xVAL (an output variable of type CHARACTER, INTEGER, or REAL, depending on whether “x” is a “C”, an “I”, or an “R”) is the name of a variable into which the value of the internal parameter specified by PNAM is to be retrieved.

- If “x” is a “C”, the internal parameter must be of type CHARACTER.
- If “x” is an “I” or an “R”, the internal parameter may be of type INTEGER or REAL; if the type of “xVAL” and the type of the internal parameter do not match, the effect will be the same as that of a mixed-mode arithmetic replacement statement in FORTRAN.

For more information, see the section “INTERNAL PARAMETERS”.

3.0 INTERNAL PARAMETERS

DASHPACK has twenty internal parameters, each of which affects, in some way, the behavior of the curve-drawing routines. The current value of a parameter may be retrieved by calling DPGETC, DPGETI, and/or DPGETR and it may be reset by calling DPSETC, DPSETI, and/or DPSETR. If an integer or real parameter is supposed to have a value in a certain numerical range and one attempts to give it a value outside that range, no error will result: the parameter will simply be given the value at the nearer end of the range.

The internal parameters of DASHPACK are as listed in the following table:

Name	Type	Description
'CRB'	C	"Character Representing Break" - the single character that is to be used in a label string in a character dash pattern to represent a break point in the label string. (Each piece of the label string is written using a separate call to PLCHHQ, PLCHMQ, or PLCHLQ; the effect is to make the label "bend" with the curve that it labels.) The default value of 'CRB' is a vertical bar.
'CRG'	C	"Character Representing Gap" - the single character that is to be used in a character dash pattern to represent a gap (of width 'WOG') in the curve. The default value of 'CRG' is an underscore.
'CRS'	C	"Character Representing Solid" - the single character that is to be used in a character dash pattern to represent a solid section (of width 'WOS') of the curve. The default value of 'CRS' is a dollar sign.
'DPL'	I	"Dash Pattern Length" - the length of the internal parameter 'DPT', as set by DPSETC or by a subsequent call to DPSETI or DPSETR. Calling DPSETC with first argument 'DPT' sets both 'DPT', which is the character dash pattern, and 'DPL', which is its length; 'DPL' can later be reset (normally to a smaller value) by a subsequent call to DPSETI or DPSETR with first argument 'DPL' , thus requesting the use of a different number of characters of the specified dash pattern. The value of 'DPL' must be between 1 and 256. The default value is 16.
'DPS'	I	"Dash Pattern Selector" - selects the dash pattern to be used. A negative value of 'DPS' says that the integer dash pattern (as specified by a call to DPSETI or DPSETR with first argument 'DPT') is to be used. If the absolute value is "n", then the low-order "n" bits of the integer are used as the dash pattern. No more than 32 bits of an integer dash pattern may be used; using more than 24 may be a problem on some systems. A positive value of 'DPS' says that the current character dash pattern (as specified by a call to DPSETC with first argument 'DPT') is to be used; 'DPS' = 0 says to use the first 'DPL' characters of the dash pattern, while 'DPS' > 0 says to use the first 'DPS' characters of the dash pattern. No more than 256 characters of a character dash pattern may be used. The value of 'DPS' must be between -32 and +256. The default value is 0.

Name	Type	Description
'DPT'	C or I	"Dash PaTtern" - one of the current dash patterns. (In a call to DPSETC or DPGETC, the name 'DPT' refers to the current character-string dash pattern, but in a call to DPSETI, DPGETI, DPSETR, or DPGETR, it refers to the current integer dash pattern. Which of these is actually in use at a given time is specified by the value of the internal parameter 'DPS'.) In an integer dash pattern, 1 bits represent solids and 0 bits represent gaps. No more than 32 bits of an integer dash pattern may be used; using more than 24 bits may be a problem on some systems. A character dash pattern is a string of 256 or fewer characters; in such a string, occurrences of the characters specified by the values of 'CRG' and 'CRS' specify gaps and solids, respectively. Other characters in the dash pattern form label strings to be written along a curve. Within each complete label string of a character dash pattern, the character specified by the value of 'CRB' may be used to specify "break points" at which the label string may be broken into smaller substrings. Alternatively, the single-character flag 'SCF' may be set non-zero to say that the label string may be broken into single-character substrings. Since each such substring is written by a separate call to PLCHHQ, PLCHMQ, or PLCHLQ, the effect of breaking up a label string is to make the label "bend" with the curve. The default character dash pattern consists of sixteen dollar signs, and the default integer dash pattern is "65535", which has sixteen low-order 1s.
'EPS'	R	"EPSilon" - says how far apart two points have to be (in X or Y, in the fractional coordinate system) in order to be considered separate points by the smoothing routine DPSMTH. The value of 'EPS' must be greater than or equal to zero. The default value is .000001.
'LS1'	R	"Label Spacing parameter 1" - specifies how much extra gap space to leave at the beginning and end of a label. Giving 'LS1' a non-zero value helps to ensure that there will be a sufficiently large gap to (for example) prevent a leading minus sign from appearing to be part of the line. The value of 'LS1' is given as a multiple of the value of the parameter 'WOC' (the width of a character); it must not be less than zero nor greater than 10. The default value is .5.
'LS2'	R	"Label Spacing parameter 2" -specifies how much extra gap space to leave for each piece of a broken label. When break characters are used or the single-character flag 'SCF' is turned on, 'LS2' determines the spacing of the characters along the line. The value of 'LS2' is given as a multiple of the value of the parameter 'WOC' (the width of a character); it must not be less than zero nor greater than 10. The default value is 0.
'LTL'	I	"Line Through Label" - a flag that says whether or not each label substring specified by a character dash pattern is to be written in a gap ('LTL' = 0) or just on top of the curve ('LTL' = 1). The latter is most effective if the line is one color and the labels are another color. The value of 'LTL' must be either 0 or 1. The default value is 0.

Name	Type	Description
'MFS'	R	“Multiplier for First Solid” - a real multiplier for the length of an initial solid portion of a curve drawn by DASHPACK. The object of using this is to make it possible to slightly offset labels on curves that are very nearly parallel to one another (as can happen, for example, when drawing contour lines). The value of 'MFS' must be greater than or equal to zero. The default value is 1.
'PCF'	I	“PlotChar Flag” - says which PLOTCHAR routine is to be called to draw character strings. The value 0 says to call PLCHHQ, the value 1 says to call PLCHMQ, and the value 2 says to call PLCHLQ. The value of 'PCF” must be either 0, 1, or 2. The default value is 0.
'SAF'	R	“String Angle Flag” - says how labels are to be oriented. If 'SAF' = 0, labels are written along a curve in the direction in which the curve is being drawn. If 'SAF' is negative, labels are written in the direction ABS('SAF') degrees, but this is done only if 'LTL' is non-zero; otherwise, the code behaves as if 'SAF' were zero: labels are written in the direction of the curve. If 'SAF' is greater than zero, labels are written along the curve, but the angle is adjusted by adding multiples of 180 degrees so that the resulting angle lies in the range from 'SAF'-90 to 'SAF'+90 degrees. If a label string is broken into substrings (either because there are “break” characters in it or because 'SCF' is non-zero), a negative value of 'SAF' will be treated as a zero value; a value greater than zero may cause the entire label to be written in the opposite direction along the curve if that ensures that more characters of the label will be written at angles between 'SAF' -90 and 'SAF'+90 degrees. Generally, when 'SAF' is non-zero, it is either -360 or +360, which has the effect of making the labels as nearly upright as possible on the frame. The value of 'SAF” must be in the range from -360 to +360. The default value is 360.
'SBF'	I	“String Buffering Flag” - a flag that says whether output of labels is to be buffered or not. When 'SBF' is non-zero, buffering is done. This ensures that, if the end of a curve occurs anywhere within a particular label, no part of the label is written; instead, that part of the curve is drawn using only the gap and solid elements of the dash pattern. Buffering is also important when 'SCF' is non-zero or there are “break” characters in a label string and 'SAF' is greater than zero; in this case, the buffering makes it possible to reorient the label as needed to make most of it upright. When 'SBF' is zero, buffering is turned off. There may be gaps at the ends of curves. If, in addition, 'SCF' is non-zero or there are “break” characters in label strings, there may be partial labels at the ends of curves. The value of 'SBF' must be either 0 or 1. The default value is 1.

Name	Type	Description
'SCF'	I	“Single Character Flag” - when 'SCF' is non-zero, it says that the label-string portions of character dash patterns are to be broken into single-character pieces, each of which is to be written by a separate call to PLCHHQ, PLCHMQ, or PLCHLQ. If 'SCF' = 0, label strings are broken into pieces only at the break points indicated by the use of “break” characters in the strings. It is not appropriate to use 'SCF' non-zero when PLCHHQ is being used and a label string in the dash pattern contains function codes that are meaningful to PLCHHQ; in that case, one should leave 'SCF' = 0 and use the “break” character 'CRB' in the label string to tell DASHPACK where it can be broken. The value of 'SCF” must be either 0 or 1. The default value is 0.
'SSL'	R	“Smoothed Segment Length” - specifies how far apart the points used to draw a smoothed curve should be. The value of 'SSL' is given in the fractional coordinate system and must be in the range from .000001 to 1. The default value is .01.
'TCS'	R	“Tension on Cubic Splines” - a value which, if negative, turns smoothing off, and which, if non-negative, turns smoothing on and, if greater than zero, specifies the desired tension to be used on the cubic splines used to do the smoothing. Note that only the routines DPCURV, DPFRST, DPVECT, DPLAST, and DPSMTH are affected by the value of 'TCS'; the routines DPLINE and DPDRAW never smooth. The routine DPSMTH always smooths: if 'TCS' is less than or equal to zero, simple cubic splines are used, and, if 'TCS' is greater than zero, splines under tension are used, in which case 'TCS' specifies the desired tension. It’s a bad idea to use values of 'TCS' much bigger than about 15, as this can cause overflows in the smoothing routines. The default value is -1.
'WOC'	R	“Width Of Character” - the character width to be used in writing labels. The value of 'WOC' is given in the fractional coordinate system and must be in the range from .000001 to 1. The default value is .01.
'WOG'	R	“Width of Gap” - the width of each gap in the dashed line. The value of 'WOG' is given in the fractional coordinate system and must be in the range from .000001 to 1. The default value is .005.
'WOS'	R	“Width of Solid” - the width of each solid in the dashed line. The value of 'WOS' is given in the fractional coordinate system and must be in the range from .000001 to 1. The default value is .005.

4.0 ERROR HANDLING

When a DASHPACK 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 DPGETR - PARAMETER NAME NOT KNOWN - QAT
```

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 ("DPGETR", in the example), a blank, a minus sign, another blank, and, lastly, a short description of the error (in this case, to the effect that a specified parameter name is unrecognized).

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 DASHPACK 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:

```
DPGETI/DPGETR - PARAMETER NAME NOT KNOWN - QAT
```

What this particular error message says is that DPGETI called DPGETR, which detected an error condition and called SETER. Upon getting control back from DPGETR, DPGETI detected the fact that DPGETR 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 DASHPACK 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 DASHPACK 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 DASHPACK routine may not be listed here, but in the programmer document for some other package.

- **DPCURV - UNCLEARED PRIOR ERROR**

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

- **DPDRAW - IMPLEMENTATION ERROR - SEE SPECIALIST**

  An internal error has occurred which indicates that DASHPACK has not been correctly installed. See a specialist.

- **DPDRAW - UNCLEARED PRIOR ERROR**

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

- **DPFRST - UNCLEARED PRIOR ERROR**

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

- **DPGETC - PARAMETER NAME NOT KNOWN - X**

  The first argument in a call to DPGETC is not one of the legal internal parameter names of DASHPACK. “X” is the value of the offending first argument.

- **DPGETC - PARAMETER NAME TOO SHORT - X**

  The given parameter name is only one or two characters long. All DASHPACK parameter names are at least three characters long, so there is something wrong. “X” is the value of the offending first argument.

- **DPGETC - UNCLEARED PRIOR ERROR**

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

- **DPGETI - UNCLEARED PRIOR ERROR**

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

- **DPGETR - PARAMETER NAME NOT KNOWN - X**

  The first argument in a call to DPGETR is not one of the legal internal parameter names of DASHPACK. “X” is the value of the offending first argument.

- **DPGETR - PARAMETER NAME TOO SHORT - X**

  The given parameter name is only one or two characters long. All DASHPACK parameter names are at least three characters long, so there is something wrong. “X” is the value of the offending first argument.

- **DPGETR - UNCLEARED PRIOR ERROR**

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

- **DPLAST - UNCLEARED PRIOR ERROR**

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

- **DPLINE - UNCLEARED PRIOR ERROR**

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

- **DPSETC - PARAMETER NAME NOT KNOWN - X**

  The first argument in a call to DPSETC is not one of the legal internal parameter names of DASHPACK. “X” is the value of the offending first argument.

- **DPSETC - PARAMETER NAME TOO SHORT - X**

  The given parameter name is only one or two characters long. All DASHPACK parameter names are at least three characters long, so there is something wrong. “X” is the value of the offending first argument.

- **DPSETC - UNCLEARED PRIOR ERROR**

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

- **DPSETI - UNCLEARED PRIOR ERROR**

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

- **DPSETR - PARAMETER NAME NOT KNOWN - X**

  The first argument in a call to DPSETR is not one of the legal internal parameter names of DASHPACK. “X” is the value of the offending first argument.

- **DPSETR - PARAMETER NAME TOO SHORT - X**

  The given parameter name is only one or two characters long. All DASHPACK parameter names are at least three characters long, so there is something wrong. “X” is the value of the offending first argument.

- **DPSETR - UNCLEARED PRIOR ERROR**

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

- **DPSMTH - UNCLEARED PRIOR ERROR**

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

- **DPVECT - UNCLEARED PRIOR ERROR**

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

5.0 EXAMPLES

On a Unix system on which NCAR Graphics has been installed, the command “ncargex” may be used to obtain and run an example showing how to use DASHPACK. The name of the example is “tdshpk”, so the command one uses is

```
ncargex tdshpk
```

To look at the resulting metafile, use the command

```
ctrans tdshpk.ncgm
```

To acquire the code for the example without running it, use the command

```
ncargex -n tdshpk
```

After the source file has been modified, compile and run it using the commands

```
ncargf77 tdshpk.f
a.out
```

Then, to look at the resulting metafile, use the command

```
ctrans gmeta
```

(For “ctrans” to work properly, one or more environment variables may need to be set.)

The unmodified example produces three frames:

- Frame 1 shows three boxes drawn using DPLINE and three ovals (actually, they’re Piet Hein super-ellipses) drawn using DPCURV.

  The first box is drawn using the default character dash pattern, which produces a solid line.

  The second box is drawn using a simple character dash pattern specifying mostly solids and a single label ('A').

  The third box is drawn using a simple 14-bit binary dash pattern.

The first oval is drawn with the smoother off and with a character dash pattern in which the label is not allowed to be broken into smaller pieces; the label on the left end of the oval is written where the curve is changing direction rapidly and it therefore obliterates much information about what the curve is doing there.

The second oval is drawn with the smoother off, but with a character dash pattern in which the label includes “break” characters. The label on the left now bends with the curve; part of it is upside down, but it is readable and it gives a good notion of what the curve is doing.

The third oval is drawn in much the same way as the second, except that the smoother is turned on and the internal parameter 'SCF' is used to say that the label can be broken anywhere, so that the “break” characters can be omitted from the label in the dash pattern. Note that the right end of the oval is drawn even though, at that point in drawing it, the end of the curve occurs in the middle of a gap being left for a label; the older dash packages would have reacted (at best) by drawing a straight line segment there.

- Frame 2 shows four spirals that are drawn in various ways:

  The spiral in the upper left quadrant is drawn using DPFRST, DPVECT, and DPLAST. Note that only one of the labels is written in the direction in which the spiral was drawn; the other two are written in the opposite direction. In all three cases, this ensures that a majority of the characters are right-side up. This is as requested by the default setting of the internal parameter 'SAF'.

  The spiral in the upper right quadrant is drawn using DPDRAW. Note that, near the outermost end of the spiral, there are two dashes that are longer than the others. This is due to the fact that, at that point, a gap was being left for another label; when the end of the curve occurred, the gap was filled, not with a label, but with a curve drawn using only the gap and solids in the dash pattern. Mostly, this works okay; if a better solution happens along in the future, I’ll implement it.

  The spiral in the lower left quadrant is just like the one in the upper right quadrant, except that it was drawn using DPSMTH instead of DPDRAW.

  The spiral in the lower right quadrant illustrates three things: 1) the use of PLCHHQ function codes to write the “times” sign and to write the “14” as a superscript; this makes it necessary to use break characters in the label to tell DASHPACK where it is safe to break the label into smaller pieces; 2) the use of the internal parameter 'LTL' to extend the curve through the label; and 3) the use of different colors for the curve and the labels.

- Frame 3 shows four spirals illustrating miscellaneous features of DASHPACK:

  The spiral in the upper left quadrant is drawn using horizontal labels. This is done by giving the internal parameter 'SAF' the value -360 and by setting 'LTL' non-zero. (Currently, DASHPACK has no way of knowing how much of a gap to leave for a horizontal label, so it insists that to get horizontal labels you must also have set 'LTL' non-zero so as to draw the curve line through them.)

  The spiral in the upper right quadrant is drawn with 'PCF' set to 1 to request the use of PLCHMQ in place of PLCHHQ. Setting 'PCF' to 2 would request the use of PLCHLQ.

The spiral in the lower left quadrant illustrates the capability of changing the sizes of the characters used for the curve labels and the lengths of the gaps and solids.

The spiral in the lower right quadrant illustrates (though perhaps not very well) the use of the internal parameter 'FSM' to offset labels on one curve from labels on an adjacent curve. I do not view this as a terribly useful feature; it’s a carryover from the older packages.

If you extract a portion of the code from this example and run it by itself, be careful to also extract all preceding statements that set internal parameters affecting that portion of the code; otherwise, you may not get exactly the same results.

6.0 MISCELLANY

DASHPACK was written in July, 1994. Some of the ideas in it were taken from the earlier dashed-line packages in NCAR Graphics.

User-level entry points in the package DASHPACK are as follows: DPCURV, DPDRAW, DPFRST, DPGETC, DPGETI, DPGETR, DPLAST, DPLINE, DPSETC, DPSETI, DPSETR, DPSMTH, and DPVECT. There is also a BLOCK DATA routine called DPBLDA.

DASHPACK uses the following common block names: DPCMCH and DPCMRI.

DASHPACK uses the routine SETER and various routines from the package SPPS.

DASHPACK is written in portable FORTRAN-77.