PLOTCHAR, A CHARACTER-PLOTTING PACKAGE

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

PLOTCHAR, A CHARACTER-PLOTTING PACKAGE

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

1.0 INTRODUCTION

This is the reference manual for an NCAR Graphics package called PLOTCHAR, which includes routines allowing a user to draw characters of high, medium, or low quality.

The high-quality routine provides many features, among which are the following: Many fonts are available, including some filled fonts. The aspect ratio of the characters can be changed at will. Characters can be written in any desired size. Characters can be written across the frame or down the frame. Subscripting and superscripting can be done easily. Character strings can be centered in various ways relative to a given point in the user coordinate system. Text-extent quantities can be retrieved to find out how much space a character string will occupy when it is finally written. Characters can be drawn with shadows and outlines; colors can be specified by the user for all parts of the characters, including the shadows and outlines. Each character string written can be enclosed in a box, which may itself have a shadow and/or an outline; colors can be specified for the parts of the box.

This document will make better sense if the reader runs the example “epltch” described in the section “EXAMPLES” and examines the resulting plots while reading. In particular, those plots will be needed in order to select unusual characters to be drawn.

2.0 OVERVIEW

All entry points in the package PLOTCHAR, whether user-callable or not, have six-character names beginning with the characters 'PC'. (The original names of the three principal routines all began with 'PLCH', but there are now alternate names for these routines that do begin with 'PC'.)

There are twelve user-callable routines in PLOTCHAR. Three of these are used to draw characters and nine are used to access internal parameters that affect the behavior of the character-drawing routines.

A thirteenth routine is normally called by PLOTCHAR itself to map character coordinates from an arbitrary X/Y coordinate system to some other system (e.g., the surface of the globe) and may be replaced by the user to implement mappings other than the default ones.

2.1 Character-Drawing Routines

Each of the FORTRAN statements

```
CALL PLCHHQ (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
CALL PLCHMQ (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
CALL PLCHLQ (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
```

draws the character string specified by the character variable CHRS at the position in the user coordinate system specified by (XPOS,YPOS). The size of the characters to be used, the angle at which they are to be written, and the way in which the whole string is to be positioned relative to (XPOS,YPOS) are specified by the arguments SIZE, ANGD, and CNTR.

Note: The new names PCHIQU, PCMEQU, and PCLOQU are just synonyms for the original names PLCHHQ, PLCHMQ, and PLCHLQ, respectively; therefore, one may also use one of the following FORTRAN statements:

```
CALL PCHIQU (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
CALL PCMEQU (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
CALL PCLOQU (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
```

PLCHHQ draws high-quality characters selected from the PWRITX database or from any of the fontcap-defined databases used by the translators. Characters from these databases can be mixed together in the graphic output resulting from a single call to PLCHHQ. The PWRITX database includes variable-width Roman and Greek fonts in three different “sizes” (“principal”, “indexical”, or “cartographic”) and in upper and lower case; many special characters are available. The fontcap databases include Roman, Greek, Cyrillic, and Gothic characters, italics, math symbols, music symbols, astronomical symbols, and miscellaneous other symbols; some of the fonts are stroked (using nothing but the GKS polyline primitive) and some are filled (using the GKS fill-area primitive). Subscripting and superscripting can be specified. Text-extent vector magnitudes can be computed and recovered to find out how large the string drawn was (or how large it will be). One has precise control over the aspect ratio of the characters and their positioning with respect to one another. Characters can be drawn with shadows and/or outlines. A box can be drawn around the character string created by a single call to PLCHHQ; the box can itself have a shadow and/or an outline. One can set the color and line width to be used while characters are drawn. One can change the order in which characters are drawn; one can, in addition, squeeze the characters together and thus effectively “stack” them from left to right or from right to left. One can request that the characters drawn be subjected to one of the mappings defined by the routine PCMPXY; the default version of this routine defines a useful set of mappings, and one can define one’s own mappings by supplying a replacement version of it. One can force characters to be drawn using PLCHMQ or PLCHLQ; the advantage of this is that subscripting and superscripting can still be done easily, but CPU time and metafile size are reduced.

PLCHMQ draws characters of medium quality. It offers upper and lower case Roman characters, as well as a number of other characters from the standard set of printable ASCII characters. The aspect ratio of the characters can be changed. Characters can be mapped using the routine PCMPXY.

PLCHLQ draws characters of the lowest (or, more accurately, the least predictable) quality. It calls the GKS primitive GTX to draw the characters, thus creating a smaller metafile, but making one dependent on capabilities of the translator.

An idiosyncrasy of PLOTCHAR is that PLCHHQ and PLCHMQ draw characters using calls to the GKS routines GPL and GFA, while PLCHLQ uses calls to the GKS routine GTX. This implies that, to change the color of the characters using GKS calls, one has to call either GSPLCI and GSFACI, in the one case, or GSTXCI, in the other. (A better way to change the color of characters drawn by PLCHHQ is to use the "internal parameters" provided for color-setting.)

Warning: PLCHHQ uses the FORTRAN logical unit specified by the value of the internal parameter 'UN' (which has the value 3, by default). Users should either avoid using unit 3 or reset the value of this internal parameter to make PLCHHQ use some other unit. The routines PLCHMQ and PLCHLQ pose no such problem.

2.2 "Internal Parameters" and Routines to Access Them

The behavior of the routine PLCHHQ (and, to a lesser extent, PLCHMQ and, to an even lesser extent, PLCHLQ) is determined not just by its arguments, but also by a set of internal variables whose values are preserved from call to call. User access to the values of these "internal parameters" is provided by the following calls:

```
CALL PCGETC (PNAM,CVAL)
CALL PCGETI (PNAM,IVAL)
CALL PCGETR (PNAM,RVAL)

CALL PCSETC (PNAM,CVAL)
CALL PCSETI (PNAM,IVAL)
CALL PCSETR (PNAM,RVAL)
```

The first argument in a call to one of these six routines specifies the name of an internal parameter (a two-character mnemonic or, sometimes, a two-character mnemonic followed by an index enclosed in parentheses) and the second is either a variable to receive the value retrieved or an expression whose value is to be stored. The last character of the routine name corresponds to the type of the second argument: Character, Integer, or Real.

Most of the internal parameters are singly-dimensioned, but a couple of them are arrays. The name of a particular element of an internal parameter array is formed by appending the index of the element, enclosed in parentheses, to the two-character name of the internal parameter array. If the index is a constant, this is easily done, but, if the index is a variable, forming the name is rather clumsy. The function PCPNWI is provided to make things a little easier for the user: The following function reference may be used to obtain the name of a specified element of an internal parameter array; the name may then be used in a call to one of the routines PCGETx or PCSETx:

```
PCPNWI (PNAM,IPAI)
```

("PNAM" is the internal parameter name and IPAI is an integer variable containing the desired index value.) Note that any routine that uses PCPNWI must contain the following declaration:

```
CHARACTER*16 PCNPNWI
```

One of the internal parameters is an array of "special" colors that may be used in drawing some characters (for example, the state highway symbol is normally drawn using a particular pair of colors). It is possible to define any desired set of colors for these, but the following call can be used to define a default "normal" set of colors for them:

```
CALL PCDLSC (IFCI)
```

The single argument specifies the first of sixteen available color indices to be defined by PCDLSC. (At the moment, such a call is somewhat pointless, because none of the currently available fonts actually uses any of the special colors; in the future, this routine will become more useful than it is now.)

A final routine is provided to reset the values of all the internal parameters to their default values. It is called using the statement:

```
CALL PCRSET
```

There are no arguments in a call to PCRSET.

2.3 The Mapping Routine

When PLCHHQ, PLCHMQ, or PLCHLQ is told to "map" characters through a transformation routine, it uses calls like

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

The routine PCMPXY is expected to transform the incoming coordinates XINP and YINP as directed by the mapping flag IMAP and return the resulting coordinates as the values of XOTP and YOTP. The default version of this routine provides for IMAP = 1 (EZMAP transformations), IMAP = 2 (rho/theta transformation), IMAP = 3 (perspective transformation from an arbitrary plane in 3-space to a projection plane, IMAP = 4 (an EZMAP transformation intended to allow one to label specific points on the globe with minimal distortion), and IMAP = 100 (provided for the sole purpose of doing clipping for the NCAR Graphics utility STITLE); all other positive values of IMAP serve to select the identity transformation. Users can supply their own versions of PCMPXY to define other mappings.

2.4 Error Conditions

An unrecognized character may result in the drawing of a blank or a five-pointed star (depending on which routine is called). There are also various error conditions which will cause PLOTCHAR to call the error-handling routine SETER. See the section "ERROR MESSAGES', below.

3.0 PLCHHQ (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)

Warning: PLCHHQ uses the FORTRAN logical unit specified by the value of the internal parameter 'UN' (which has the value 3, by default). Users should either avoid using unit 3 or reset the value of this internal parameter to make PLCHHQ use some other unit.

PLCHHQ draws high quality characters. By default, it uses the same database as the old NCAR Graphics routine PWRITX, but it can also use characters from any of the fontcap-defined databases, it has an improved interface, and it has many more capabilities than PWRITX.

Note: The name PCHIQU, which better fits the naming pattern used for other routines in PLOTCHAR, may now be used as a synonym for PLCHHQ.

Either of the FORTRAN statements

```
CALL PLCHHQ (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
CALL PCHIQU (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
```

writes the string of characters specified by the FORTRAN character string CHRS at the position (XPOS,YPOS). The size of the characters, the angle at which they are written, and the way in which the string is positioned relative to (XPOS,YPOS) are specified by the arguments SIZE, ANGD, and CNTR.

The remainder of this section is divided into the following subsections:

- The various character sets available to PLCHHQ are described in "The 'Complex' Character Set", "The 'Duplex' Character Set", and "Other Character Sets".
- Various features available to the user of PLCHHQ are described in "Color, Shadows, Outlines, Boxes, and Character Stacking", "The 'Quality' Flag of PLCHHQ', "Mapping Characters from One X/Y Coordinate Space to Another", and "Miscellaneous Other Internal Parameters".
- Detailed descriptions of PLCHHQ's arguments are given in "Arguments".
- Detailed descriptions of the "function codes" which are used for various purposes in the character string CHRS are given in "Function Codes - Details".
- Detailed descriptions of internal parameters that affect the behavior of PLCHHQ are given in "Internal Parameters - Details".

3.1 The "Complex" Character Set

By default, PLCHHQ draws characters from a subset of the PWRITX database, called the "complex" character set, that contains 564 different characters. Obviously, since the printable ASCII and EBCDIC character sets are not that large and the set of standard FORTRAN characters is even smaller, there cannot be a one-to-one mapping between the characters in CHRS and the characters drawn. The scheme that allows one to select any of the 564 characters in the PWRITX database using only legal FORTRAN characters in CHRS is described in the next paragraph. It is recommended that, before proceeding, the reader obtain the plots produced by the example “epltch”, which is described in the section called “EXAMPLES”; they will make it easier to follow the discussion of the character-selection scheme.

The 564 characters of the complex character set are divided into 12 groups of 47 characters apiece. Each of the 12 groups is characterized by font (Roman or Greek), size[1] (principal, indexical, or cartographic), and case (upper or lower). Within each group, each character is associated with one of the 47 characters A-Z, 0-9, +, -, *, /, (,), $, =, blank, comma, and period. Initially, as the scan of the character string CHRS starts, the font, size, and case are assumed to be Roman, principal, and upper, respectively. Thus, if CHRS = 'A', a Roman A, of principal size, in upper case, will be drawn. As the scan proceeds from left to right, the font, size, and case can be changed by the occurrence of “function codes”, which are just characters enclosed by pairs of function code signal characters, as defined by the value of the internal parameter 'FC' (the default value is a colon, but this can easily be changed by the user). For example, if CHRS = ':L:A', a Roman A, of principal size, but in lower case, will be drawn (because the “L” is a function code used to switch to lower case). If CHRS = ':IGL:A', an indexical-size, lower-case Greek alpha will be drawn (because the “I” is a function code used to switch to indexical size and the “G” is a function code used to switch to the Greek font).

Using function codes for font, size, and case allows all of the 564 characters in the complex character set to be accessed using nothing but the 48 standard FORTRAN characters.

Forty-five other characters can appear in the character string CHRS. These include the 26 lower-case alphabetic characters and the 19 special characters exclamation point, double quote, pound sign, percent sign, ampersand, colon, semicolon, less than, greater than, question mark, at sign, left square bracket, backward slash, right square bracket, left curly bracket, vertical bar, right curly bracket, tilde, and apostrophe. Each of these results in the selection of the logical character from the complex character set. Note that the use of these characters must be viewed with some suspicion, since the FORTRAN-77 standard indicates that the effect of using them in a character string is “processor dependent”; however, I would be very surprised to find a system on which using them caused other than the desired behavior.

Three other ASCII characters - the hat, the underscore, and the backwards apostrophe - are recognized by PLCHHQ, but, since these characters do not exist in the complex character set, each will cause the drawing of a five-pointed star. (The same five-pointed star will be drawn for totally unrecognized ASCII characters.)

1. The word “size” is probably somewhat misleading here. Principal, indexical, and cartographic characters do differ in digitized size, but they also differ in shape and really constitute different fonts. The final size of the characters written is determined as much by the PLCHHQ argument SIZE as it is by the function codes setting “size”. The terminology was invented for the PWRITX package and is perpetuated here.

3.2 The “Duplex” Character Set

PLCHHQ can be made to use a different 564-character subset of the PWRITX database, called the “duplex” character set. Duplex characters are simpler in appearance (no serifs, fewer curves, etc.) but are still of high quality. (Only the alphanumeric characters in the two sets are different; all other characters are the same.) To make PLCHHQ access the duplex set, simply call PCSETI to reset the internal parameter 'CD' to 1 instead of its default value, 0. To return to the complex set, just reset 'CD' to 0 again. Be aware that each switch from one set to the other requires that PLCHHQ read a binary file the next time it is called. The method of selecting a particular character from the duplex set (using function codes to set font, case, and size and then one of 47 standard FORTRAN characters to select a specific character) is the same as for the complex set.

3.3 Other Character Sets

PLCHHQ can be made to use characters from other character databases (as of 12/92, 43 of them); each such database is defined by what is called a “fontcap” and is normally used by the metafile translators. The internal parameter 'FN' (for “Fontcap Number) has the default value 0. If it is given a non-zero value “n”, characters from fontcap database “n” are used instead of characters from the PWRITX database. 'FN' can be set either by calling PCSETI with “n” as the second argument or by calling PCSETC with the name of the desired font as the second argument. (In the subsection “Internal Parameters - Details”, there is a table showing the correspondence between font numbers and font names.)

It is possible to switch from one fontcap database to another (or to resume the use of characters from the PWRITX database) by means of “F” function codes in CHRS. For example, if the current value of 'FN' is 13 and CHRS = 'A:F1:B:F2:C:F0:D:F:E', the effect will be to select the character from fontcap database 13 that is associated with an “A”, followed by the character from fontcap database 1 that is associated with a “B”, followed by the character from fontcap database 2 that is associated with a “C”, followed by an upper case Roman “D” of principal size from the PWRITX database (which is thought of as “font” 0), followed by the character from fontcap database 13 that is associated with an “E” (“F” without a following integer says to go back to the font defined by the current value of 'FN'). Keep in mind that each occurrence of an “F” or “Fn” function code may result in reading a binary file.

Each of the fontcap databases defines at most 128 characters (usually no more than 97). Desired characters are selected by just using the appropriate ASCII characters in CHRS. For example, to write the character associated with an upper case “A”, just use the string 'A', and to write the character associated with a lower case “A”, just use the string 'a'.

The example “epltch”, described in the section “EXAMPLES”, produces a frame showing all of the characters defined by the fontcap databases and shows how to access each of those characters.

Function codes that alter the “font” (“R” for “Roman” and “G” for “Greek”) for PWRITX database access have no effect on fontcap database access. Function codes that force the use of lower case (“L” for “Lower”) will cause upper-case alphabetic characters to be treated as if they were lower-case in selecting characters from fontcap databases. Function codes that alter “size” (“P” for Principal, “I” for “Indexical, and “K” for “Cartographic”) will cause a similar shift in the sizes of characters used from fontcap databases.

3.4 The Filled Fonts

In version 3.2 of NCAR Graphics, eleven filled fonts were introduced. Use of a filled font is requested in the same way as any of the other fontcap-defined fonts: by setting the value of the internal parameter 'FN' to the number of the desired font (using a call to PCSETI) or to the name of the desired font (using a call to PCSETC) or by using function codes of the form “Fn”, where “n” is the desired font number.

Medium to large characters from the filled fonts look much better than those from the other fonts, because there are no individual strokes to become visible as the characters are enlarged. Small characters from the filled fonts may not look as good as those from the other fonts, however - particularly on a low-resolution device.

The default value of the GKS parameter for “fill area interior style” implies “hollow” fill, which is not terribly useful. If PLCHHQ sees that the current value of this parameter is “hollow”, it will instead use “solid”; however, if the current value is “pattern” or “hatch”, that value will be used. In any case, upon return from PLCHHQ, the value of the parameter will have been restored to what it was on entry.

For each filled font, there is an outline font; the number of each outline font is 100 more than the number of the filled font and the name of each outline font is the same as that of the filled font with a prepended “O_”.

3.5 Color, Shadows, Outlines, Boxes, and Character Stacking

The internal parameters 'CC' (“character color”) and 'CL' (“character line width”) specify the principal color and line width to be used when drawing characters.

The internal parameter 'SF' (the “shadow flag”) determines whether or not character shadows are drawn. The shadow of a character is just a copy of the character, drawn in a single color prior to drawing the character itself, with slight offsets in the directions ANGD and ANGD+90. Shadows can be used with any font, but they look best when used with the filled fonts. The internal parameters 'SC', 'SL', 'SX', and 'SY' specify the color, the line width, and the dimensions of the offsets in the two directions, respectively.

The internal parameter 'OF' (the “outline flag”) determines whether or not character outlines are drawn. For non-filled fonts, the outline of a character is just a copy of the character. Each of the filled fonts has a corresponding outline font containing outlines for the filled characters. The internal parameters 'OC' and 'OL' control the color and the line width of the outlines.

The internal parameter 'BF' (the "box flag") determines whether or not a box is drawn around/behind each character string written by a call to PLCHHQ. The internal parameters 'BC', 'BL', 'BM', 'BX', and 'BY' determine the color, line width, margin width, and box shadow offsets, respectively.

The internal parameter 'DO' (for "drawing order") determines whether the characters are drawn in their order of appearance in CHRS or in the reverse order. It also determines whether one draws all shadows and then all principal character bodies and then all character outlines, or whether one draws all three parts of the first character, then all three parts of the second character, then all three parts of the third character, and so on. This parameter can be used to effect "stacking" of characters.

Note: On some devices (for example, on a film-exposing device like the Dicomed), shadows, outlines, boxes, and/or stacking must be done in such a way that, where things overlap, an overlapping color replaces an underlying color. (On NCAR's Dicomed, one uses the option "QUAL=BEST" in order to process the metafile through a frame buffer.)

3.6 The "Quality" Flag

The default value of the internal parameter 'QU' (the "quality flag") is zero. If 'QU' is set to one, individual characters are drawn, one at a time, by calling the medium-quality character generator PLCHMQ (which see); if 'QU' is set to 2, the low-quality character generator PLCHLQ is called. The advantages of this (as opposed to calling PLCHMQ or PLCHLQ directly) are that function codes used to change size and case and to do subscripting and superscripting may be used and the internal parameters that set color and request shadows, outlines, and boxes will be honored. Obviously, if none of these features is being used and lower-quality characters are acceptable, it's better to call PLCHMQ or PLCHLQ directly.

PLCHMQ uses a 94-character digitized set that is not as fancy as the PLCHHQ set, so the resulting metafile will be about half as large.

PLCHLQ draws characters using GKS primitives. The resulting metafile is considerably shortened because only the characters are sent to it, instead of all the pen moves required to stroke them out; this has the disadvantage that one then becomes dependent on capabilities of the translator that may or may not be present. At NCAR, PLCHLQ is intended to produce characters as good as those produced by PLCHMQ, but this situation cannot be guaranteed elsewhere.

3.7 Mapping Characters from One X/Y Coordinate Space to Another

The default value of the internal parameter 'MA' is zero, which says that no mapping of characters is to be done. It this parameter is given a non-zero value, it indicates that characters are to be mapped from the X/Y coordinate system in which they are positioned to some other X/Y coordinate system. This is done by calling the routine PCMPXY for each coordinate pair used in drawing the characters. The default version of PCMPXY defines a couple of useful mappings; it can be replaced by a version that defines mappings of the user's own design.

When mapping is being done, the internal parameter 'OR' becomes important. It can be given a non-zero value to specify what value will be returned by PCMPXY for points that are invisible under the selected mapping. For example, when the orthographic projection of EZMAP is being used, half of the globe is invisible and the EZMAP transformation routines MAPTRA and MAPTRN return a value of 1.E12 for the X and Y coordinates corresponding to a latitude and longitude on the invisible hemisphere. PCMPXY may do something similar; if it does, the value to be used must be made known to PLOTCHAR by setting the value of 'OR'. Characters that are partly visible and partly invisible are clipped by PLCHHQ.

When mapping is turned on, the meanings of the arguments XPOS, YPOS, and SIZE are changed somewhat. See the descriptions of these arguments for details.

3.8 Miscellaneous Other Internal Parameters

The internal parameters 'AS' (“add space”), 'CS' (“constant space), and 'SS' (subtract space) are used to change the spacing between characters.

The internal parameter 'CE' is set non-zero to select a string-centering method that leads to character strings being more accurately centered on a given point. See the discussion of the argument “CNTR”, below.

The internal parameter 'FB' is used to change the fidelity of the Bezier curves used to draw the characters in the filled fonts.

The internal parameter 'SA' is a sort of “fudge factor”, introduced in version 3.2 of NCAR Graphics and affecting the sizes of all characters written directly by PLCHHQ. Its default value is such as to make the sizes of PLCHHQ characters consistent with the sizes of PLCHMQ characters. Some users may wish to change the value of this parameter to make PLCHHQ work more nearly in the way it did in older versions of NCAR Graphics.

The internal parameter 'TE' can be used to turn on text-extent computations. Subsequently, to find the dimensions of a “text-extent box” for the last string seen by PLCHHQ, one can retrieve the values of the internal parameters 'DL', 'DR', 'DB', and 'DT'; these are directed distances from the positioning point for the string to the left edge, the right edge, the bottom edge, and the top edge, respectively, of the box. The text-extent box is just a bounding box for all the line segments used to draw the character string. The edges of the box are parallel to the edges of the string, rather than to the edges of the plotter frame, and “left”, “right”, “bottom”, and “top” are defined with reference to the string, not with reference to the plotter frame. This can be used for various purposes (e.g., when it is desired to scale a line of text to exactly fit in a given space). When 'MA' is zero (mapping is turned off), 'DL', 'DR', 'DB', and 'DT' are returned in the fractional coordinate system; otherwise, they are in the same coordinate system as that in which the arguments XPOS and YPOS are given, prior to the mapping, if any, implied by the value of 'MA'. See also the description of the argument ANGD, which can be given a special value (360.) to suppress drawing of characters, so that text-extent computations can be done for a string without actually drawing it.

The internal parameters 'XB', 'XC', 'XE', 'YB', 'YC', and 'YE' allow one to retrieve the coordinates of various parts of the last string written. These can be used for various purposes (e.g., when it is desired to draw a line from one end of the string to some feature that it serves to label or to add more characters to the string with another call to PLCHHQ). When 'MA' is zero (mapping is turned off),these parameters are returned in the fractional coordinate system; otherwise, they are in the same coordinate system as that in which the arguments XPOS and YPOS are given, prior to the mapping implied by the value of 'MA'.

The internal parameters 'ZX', 'ZY', and 'ZZ' allow one to specify the default "zooming" of characters (horizontally, vertically. or both), that applies at the beginning of a character string, prior to, or in the absence of, any occurrence of "X", "Y", or "Z" function codes. .

3.9 Arguments

As was stated above, PLCHHQ is called using either of the following statements:

```
CALL PLCHHQ (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
CALL PCHIQU (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
```

XPOS and **YPOS** are input values of type REAL, specifying positioning coordinates for the characters to be drawn. If the internal parameter 'MA' has the value 0, these are given in the current user coordinate system; otherwise, they are given in an arbitrary X/Y coordinate system, as implied by the value of 'MA' and the nature of the routine CPMPXY. (For example, if 'MA' has the value 1 and the default version of CPMPXY is being used, then XPOS is a longitude, in degrees, and YPOS is a latitude, in degrees.) The argument CNTR (described below) specifies how the characters are to be positioned relative to the point (XPOS,YPOS).

CHRS is of type CHARACTER. The number of characters in CHRS is taken to be LEN(CHRS); to use characters "m" through "n" from a character variable CHRV, use the FORTRAN-77 substring notation "CHRV(m:n)". CHRS may contain characters to be drawn and function codes; the latter are described in the section "Function Codes", below.

SIZE specifies the desired character size. If the internal parameter 'MA' is zero, then the following comments apply:

- If SIZE is less than or equal to zero, its absolute value specifies the size as a multiple of the digitized size. On a 1024x1024 grid, principal-size capitals are about 'PW' units wide by 'PH' units high and the vertical spacing between lines is 'PV' units (by default, 'PW', 'PH', and 'PV' are 16, 21, and 32). Indexical-size capitals are 'IW' units wide by 'IH' units high and the vertical spacing between lines is 'IV' units (by default, 'IW', 'IH', and 'IV' are 12, 13, and 20). Cartographic-size capitals are 'CW' units wide by 'CH' units high and the vertical spacing between lines is 'CV' units (by default, 'CW', 'CH', and 'CV' are 8, 9, and 14).

  Example: Using CHRS = 'ABC' and SIZE = -2. will write three characters, each of which is about 32/1024 of the plotter frame in width.

- If SIZE is greater than zero, but less than one, it specifies the desired approximate width of a principal-size capital as a fraction of the distance across the plotter frame. This is the recommended scheme.

  Example: Using CHRS = 'ABC' and SI ZE = .05 will write three characters, each of which is about 5 hundredths of the plotter frame in width.

- If SIZE is greater than or equal to one, it specifies the desired approximate width of a principal-size capital in plotter coordinates, as defined by default or by a user call to the SPPS routine SETI. Note that use of the routine SETI is now discouraged.

  Example: Using CHRS = 'ABC' and SIZE = 13. (assuming there have been no calls to the SPPS routine SETI) will write three characters, each of which is about 13/1024 of the plotter frame in width.

If 'MA' is non-zero, then SIZE is the desired approximate width of a principal-size capital as a value in the X/Y coordinate system in which XPOS and YPOS are given.

In most of the databases used by PLCHHQ, the character width varies from character to character. Most principal-size capitals will be a little wider than the value specified by SIZE (and a few will be narrower). Thus, a little experimenting may be necessary to get just the size one wants. A rule of thumb that works pretty well when using all capitals is as follows:

```
SIZE = DLOS / REAL(NCIS)
```

where “DLOS” is the desired length of the string, in the fractional system, and “NCIS” is the number of characters in the string.

Note: As of version 3.2 of NCAR Graphics, the size of all characters written by PLCHHQ has been reduced by a multiplicative factor (the internal parameter 'SA') whose default value is .88888888... The object of this is to make the size of characters written by PLCHHQ consistent with the size of characters written by PLCHMQ and PLCHLQ. If the value of 'SA' is not changed, then the height of principal-size capitals written by PLCHHQ and PLCHMQ will be 7/6 times the value specified by SIZE. Thus, if the desired character height (CHRH) is known, use “SIZE = 6.*CHRH/7.” .

The values of the internal parameters 'PW', 'PH', 'PV', 'IW', 'IH', 'IV', 'CW', 'CH', and 'CV' (as well as two others, 'PS' and 'IS') may be changed by user calls to CPSETR. This can be used to change the effective size and shape of digitized characters in the PWRITX database (the effect is the same as if the characters had been digitized differently at the outset). If, for example, one were to double the value of 'PH', it would make the principal characters twice as tall as they are by default. (In this case, one would probably also want to double the value of 'PV', so as to make the vertical spacing between lines consistent with the character height.) Note that, if one changes the value of 'PW', 'IW', or 'CW', the meaning of a negative value of the argument SIZE may thereby be changed.

ANGD is the angle, in degrees counterclockwise from the positive X axis, at which the character string is to be written. If the internal parameter 'TE' is non-zero (by default, it is zero) and if ANGD is exactly 360., then no characters are drawn by PLCHHQ; it just computes the distances, in the fractional coordinate system, from the point (XPOS,YPOS) to the left edge, the right edge, the bottom edge, and the top edge of a box enclosing the string. These are stored as the values of the parameters 'DL', 'DR', 'DB', and 'DT' and may be retrieved by calls to PCGETR.

CNTR is the centering option. If the internal parameter 'CE' is zero (the default), then

- CNTR = -1. means that (XPOS,YPOS) is the center of the left edge of the first character (if that character is to be written “across” the frame) or the center of the top edge of the first character (if that character is to be written “down” the frame).
- CNTR = 1. means that (XPOS,YPOS) is the center of the right edge of the last character (if the writing direction at the end of the character string is “across” the frame) or the center of the bottom edge of the last character (if the writing direction at the end of the character string is “down” the frame).
- CNTR = s, where “s” is any real number, means that (XPOS,YPOS) is a point obtained by linear interpolation along the line joining the two points mentioned above (the first point being associated with the value -1. and the second with the value +1.). The value “0.” gives the midpoint of the line.

If 'CE' is non-zero, then the value of CNTR is ignored. Text-extent quantities are computed and used to exactly center the output on the point (XPOS,YPOS). This is useful, among other things, for labelling each of a number of points with a single character; however, it works as desired only when 'QU' is zero.

Upon return from PLCHHQ, all arguments are unchanged.

3.10 Function Codes - Details

This section describes in detail the function codes that may be used in the character string CHRS. (Some of these have already been mentioned above.). Function codes are used for the following purposes:

- To select characters from the various fonts.
- To select the “level” on which characters are to be written: superscript level, subscript level, or normal level. Subscripting and superscripting may be carried to as many as five levels, as in “A to the B to the C to the D to the E to the F”. Changing level usually also changes font, case, and size (if the PWRITX database is being used) or just size (if a fontcap database is being used), but all such changes may be overridden by function codes forcing the desired values.)
- To take complete control of the positioning of characters relative to one another, and to write down, instead of across, the page.
- To specify individual characters in the PWRITX database directly, using octal numbers as function codes.
- To “zoom” characters in either width or height, or both.

As the characters of CHRS are scanned from left to right, the scanner is always in one of two states: either it is looking for characters to be drawn or it is looking for characters to be interpreted as function codes. As the scan begins, the scanner is looking for characters to be drawn. Each occurrence of the function-code signal character (defined by 'FC' - a colon by default) flips the state of the scanner. Thus, in the character string 'ABC:L:DEF', “A”, “B”, and “C” will be treated as characters to be drawn, “L” will be treated as a function-code character, and “D”, “E”, and “F” will be treated as characters to be drawn. (In this particular example, since the function code “L” requests lower case, what would be drawn is the string “ABCdef”.)

Note: Apostrophes are a poor choice for the function-code control characters. If you must use apostrophes, be aware of the FORTRAN-77 conventions for defining strings containing apostrophes (e.g., the string “What’s up, doc?”, must be written in FORTRAN as

```
'What''s up, doc?'
```

in which two apostrophes are required to represent the single apostrophe one wants).

Another example of a string containing function codes is as follows:

```
':PRU:A:S:2:N:+B:S:2:N:'
```

This might be read by a knowledgeable user as “Principal Roman Upper A, Superscript level, 2, Normal level, plus B, Superscript level, 2, Normal level” and it writes the equivalent of “A squared plus B squared”. Because of the way the colons are paired, the characters P, R, U, S, N, S, and N are function codes; all other characters are characters to be written.

No punctuation is needed between function codes except for a comma or a blank between adjacent numbers; however, extra commas and blanks may be used to improve readability. The function codes described below are the only legal ones; any other characters in a function-code string will be ignored, but an error message will be printed.

3.10.1 Database selection

These function codes determine from what database characters are to be selected:

- **F** or **F0** or **Fn**, where n is one of the legal font numbers.

The default at the beginning of a string is to use the database implied by the internal parameter 'FN'. (By default, 'FN' = 0, selecting the PWRITX database.) Fn selects the database defined by fontcap n. F0 is used to return to the PWRITX database and F is used to return to the database specified by the internal parameter 'FN'.

3.10.2 Font definitions

These function codes affect the selection of characters from the PWRITX database:

- **R** selects the Roman font.
- **G** selects the Greek font.

The default at the beginning of a string is to use the Roman font.

3.10.3 Size definitions

These function codes affect the selection of characters from the PWRITX database:

- **P** selects characters of principal size.
- **I** selects characters of indexical size.
- **K** selects characters of cartographic size.

The default at the beginning of a string is to use characters of principal size. Changing the "size" using one of these function codes really selects a different font; the terminology was used for PWRITX and is perpetuated here.

Using these function codes when characters are being selected from fontcap databases will affect the size of the characters written. For example, if CHRS = ':F16K:ABC', the characters drawn will be those defined by fontcap 16 corresponding to the upper case characters "A", "B", and "C", but they will be reduced in size to approximately match the size of the cartographic characters from the PWRITX database.

3.10.4 Case definitions

These function codes affect the selection of characters from the PWRITX database.

- **U** or **Un** selects upper case.
- **L** or **Ln** selects lower case.

If U is followed by a number n (with no intervening comma), then n characters will be drawn in upper case, and subsequent characters will be in lower case. (The Un option is particularly useful for capitalizing sentences.)

If L is followed by a number n, then n characters will be drawn in lower case, and subsequent characters will be in upper case.

The default at the beginning of a string is upper case.

Using an "L" function code will also affect the selection of characters from fontcap databases. For example, if CHRS = ':F16L:ABC', the characters drawn will be those defined by fontcap 16 corresponding to the lower case characters "a", "b", and "c".

3.10.5 Direction definitions

These function codes determine where each character is placed relative to the previous one:

- **A** causes characters to be written "across" the frame.
- **D** or **Dn** causes characters to be written "down" the frame.

The default at the beginning of a string is to write across the frame.

If D appears without a following integer “n” or if “n” is zero, characters will be written “down” until an “A” is encountered. If “n” appears, “n” characters will be written “down” and subsequent characters will be written “across” the frame.

The terms “across” and “down” are to be interpreted to mean “in the direction ANGD” and “in the direction ANGD-90”, respectively. Each character has a left edge center point, a right edge center point, a bottom edge center point, and a top edge center point, as defined by information in the database from which it came. These points are not necessarily on the actual edges of the character, but on the edges of a rectangle containing the character; the points are also not necessarily exactly centered on the edges of the rectangle. Characters are always oriented so that the vector from the left edge center point to the right edge center point is in the direction ANGD. When characters are written across the frame (assuming that the constant spacing parameter 'CS' is zero), each character is placed so that its left edge center point is coincident with the right edge center point of the previous character. Similarly, when characters are being written down the frame, each character is placed so that its top edge center point is coincident with the bottom edge center point of the previous character.

3.10.6 Level definitions

These function codes determine the level on which characters are to be written:

- **B** or **Bn** selects subscript level.
- **S** or **Sn** selects superscript level.
- **E** means “end of subscripting or superscripting”.
- **N** causes a return to normal level.

When subscripting or superscripting using characters from the PWRITX database, the character size will change depending on the base character. Principal base characters are subscripted or superscripted with indexical characters, with a shift of 'PS' digitization units up or down (10, by default). Indexical or cartographic base characters are subscripted or superscripted with cartographic characters, with a shift of 'IS' units up or down (7, by default). The case of the indexing characters will generally be the same as that of the base character. An exception is that a lower case indexical base will be subscripted or superscripted with upper case cartographic, as the cartographic size has no lower case alphabetic or numeric characters available.

When subscripting or superscripting using characters defined by the fontcap databases, a similar size change will be forced, but case will be unaffected.

Automatic size and case changes can, of course, be overridden. For example, the string 'A:SPU:B' would be used to write “A to the B”, with both A and B being of principal size and in upper case.

If S or B is followed by a positive integer n, then n characters will be drawn as specified above, after which size, case and position will be reset to that of the base character. Do not overlap level definitions given for a specified number of characters (as, for example, in the string ':S3:A:B2:BC', in which the three-character count for superscripting is not satisfied at the time the two-character count for subscripting is seen).

The function code E terminates the current level of subscripting or superscripting, restoring the size, case, and position to that of the base character.

The function code N terminates the current level of subscripting or superscripting, restoring the size and case to that of the base character, but the position as required to continue writing at the level of the base character without writing over the subscripts or superscripts.

Examples:

To write “10 to the 10 to the 100”, use the string '10:S:10:S:100'. Up to five levels are allowed.

To write “X sub 2, cubed”, use the string 'X:B1:2:S1:3'. Note that when the character count in the “B1” is satisfied, the original base character is reinstated, so that we can add another subscript or superscript to it.

To write “X sub 2, cubed, times Y cubed”, use the string 'X:B1:2:S:3:N:Y:S:3'. The function code N returns to normal level in such a way that the character Y does not overwrite the 2 and the 3.

To write “X to the power A sub 1”, followed by “something else”, use the string 'X:S:A:B:1:NN:something else'. Note that two function code N’s are used, one to terminate writing the subscript 1 and another to terminate writing the superscript “a sub 1” and leave the current position such that what follows won’t go on top of the “a sub 1”.

3.10.7 Coordinate definitions

These function codes adjust the current position, so that the next character will be drawn offset from the position it would otherwise have had:

- **Hn** or **HnQ** causes a “horizontal” move.
- **Vn** or **VnQ** causes a “vertical” move.
- **C** causes a “carriage return” before the next character is drawn.

An Hn or HnQ increments the current position in the direction specified by ANGD. Hn shifts the position by n digitization units and HnQ shifts the position by n blank widths; in either case, n can be negative.

A Vn or VnQ increments the current position in the direction specified by ANGD+90. Vn shifts the position by n digitization units and VnQ shifts the position by n blank heights; in either case, n can be negative.

A C causes a return to either the initial position, if there have been no other C function codes, or to the position following the last occurrence of a C function code, otherwise; then, the position is offset by one blank height in the direction ANGD-90. For example, the string 'ABC:C:DEF' causes the characters ABC to be written and then causes the characters DEF to be written directly under the ABC.

3.10.8 Character zooming

These function codes may be used to alter the aspect ratio of subsequent characters:

- **Xn** or **XnQ** causes character width to zoom to "n" percent of normal.
- **Yn** or **YnQ** causes character height to zoom to "n" percent of normal.
- **Zn** or **ZnQ** causes character width and height to zoom to "n" percent of normal.

Using Xn or XnQ will cause subsequent characters to be made n/100 times as wide (zoomed in the direction ANGD). A suffixed Q is ignored. If "n" is omitted or zero, 100 is assumed.

Using Yn or YnQ will cause subsequent characters to be made n/100 times as high (zoomed in the direction ANGD+90). A suffixed Q will cause a shift of character position in the direction ANGD+90 sufficient to keep the bases of the characters properly aligned. If "n" is omitted or zero, 100 is assumed.

Using Zn or ZnQ will cause subsequent characters to be made n/100 times as wide and high (zoomed in both of the directions ANGD and ANGD+90). A suffixed Q will cause a shift of character position in the direction ANGD+90 sufficient to keep the bases of the characters properly aligned. If "n" is omitted or zero, 100 is assumed.

3.10.9 Direct character access

Octal numbers may be used as function codes to select particular characters from the PWRITX database. The octal number for a given character is the sum of a font index (0 for Roman or 600 for Greek), a size index (0 for Principal, 200 for Indexical, or 400 for Cartographic), a case index (0 for Upper or 100 for Lower), and the octal equivalent of a character index (1-32 for A through Z, 33-44 for 0 through 9, or 45-57 for the individual characters +, -, *, /, (,), $, =, blank, comma, and period). For example, the string ':GIU:*' is the same as the string ':1047:' (600+200+0+47=1047, octal.) To know what character PLCHHQ will produce in response to a given octal code, refer to the plots produced by the example "epltch" described in the section "EXAMPLES".

Note: To write two of the characters described in the example in the previous paragraph in a row, use either the string ':GIU:**' or the string ':1047,1047:'. The latter is an example of the need for a comma between two function codes.

An octal number may be used as a function code in part of the input character string during the processing of which characters are being selected from one of the fontcap-defined databases. Such an octal function code will still have the effect of selecting a character from the PWRITX database.

3.11 Internal Parameters - Details

The following table shows all of the internal parameters that affect the behavior of PLCHHQ. Each entry includes the name of a parameter, its basic type, its default value, and a short description of the parameter.

TABLE 1. Internal Parameters of PLCHHQ.

Name	Type	Default	Description
'AS'	REAL	0.	When this parameter is non-zero, it forces extra space between characters. A negative value is interpreted in digitization units. A positive value is interpreted as a multiplier of the nominal character width specified by the argument SIZE (and, in the case of the PWRITX database, one of 'PW', 'IW', or 'CW', whichever is appropriate).
'BC'	INTEGER ARRAY	all -1's	An array of three "box colors". Element 1 applies to the outline of the box, element 2 to fill of the box, and element 3 to fill of the box shadow. The value "-1" means "unspecified"; and a value greater than or equal to 0 selects a particular color index. To access element 1, use the parameter name 'BC(1)'; to access element 2, use 'BC(2)'; etc. The name 'BC' by itself refers to element 1.
'BF'	INTEGER	0	When this parameter is non-zero, it requests a box around each string of characters written by PLCHHQ. The value is interpreted as a three-bit flag: Add 1 to the value to have the outline of the box drawn, 2 to have the box filled, and 4 to have its shadow drawn. (1 implies "draw outline", 2 implies "fill box", 3 implies "fill box and draw outline", 4 implies "fill shadow", 5 implies "fill shadow and draw outline", 6 implies "fill shadow and fill box", and 7 implies "fill shadow, fill box, and draw outline".)
'BL'	REAL	0.	When this parameter is greater than zero, it specifies the line width to be used while drawing the box around a character string. A value less than or equal to zero means "unspecified"; a value greater than zero specifies the desired line width as a multiplier of the "normal" line width.
'BM'	REAL	.15	The width of box margins (the space between the characters and the outline of the box), stated as a fraction of principal character height.
'BX'	REAL	-.05	The X offset for box shadows, stated as a fraction of principal character height. This is actually an offset in the direction ANGD.
'BY'	REAL	-.05	The Y offset for box shadows, stated as a fraction of principal character height. This is actually an offset in the direction ANGD+90.

TABLE 1. Internal Parameters of PLCHHQ.

Name	Type	Default	Description
'CC'	INTEGER ARRAY	all -1's	An array of color indices. Element 0 specifies the principal character color and elements 1 through 16 specify special colors to be used for certain characters. The value -1 means “unspecified”; a value greater than or equal to zero is a particular color index to be used. The routine PCDLSC may be called to define a standard default set of colors for elements 1 through 16. To access element 0, use the parameter name 'CC(0)'; to access element 1, use 'CC(1)'; etc. The name 'CC' by itself refers to element 0.
'CD'	INTEGER	0	The value 0 selects the complex dataset from the PWRITX database; 1 selects the duplex dataset.
'CE'	INTEGER	0	The value 0 selects centering of strings as specified by the argument CNTR and 1 selects exact centering, using text-extent quantities. This works as desired only when 'QU' is zero.
'CH'	REAL	9.	Digitized cartographic height.
'CL'	REAL	0.	If greater than zero, this is the desired principal line width to be used while drawing characters. A value less than or equal to zero says that line width is unspecified; a value greater than zero specifies the desired line width as a multiplier of the “normal” line width.
'CS'	REAL	0.	Constant-spacing flag. The high-quality character sets normally have variable spacing when written “across” the frame. When this parameter is set non-zero, it forces the centers of the characters to be spaced a constant distance apart. If a negative value is used, its absolute value specifies the spacing in digitization units. If a positive value is used, it is interpreted as a multiplier of the nominal character width specified by the argument SIZE (and, in the case of the PWRITX database, one of 'PW', 'IW', or 'CW', whichever is appropriate). Medium- and low-quality characters are not affected by this parameter.
'CV'	REAL	14.	Digitized cartographic vertical spacing.
'CW'	REAL	8.	Digitized cartographic width.
'DB'	REAL	none	Output only - the directed distance from (XPOS,YPOS) to the bottom edge of the text-extent box for the last string, measured in the direction ANGD-90. A negative value means that the bottom edge is in the direction ANGD+90 from (XPOS,YPOS). If the last string was effectively blank, 'DB' will be -1E6. If 'MA' is zero, 'DB' is a value in the fractional coordinate system; otherwise, it is a value in the same system in which the argument YPOS was given.

TABLE 1. Internal Parameters of PLCHHQ.

Name	Type	Default	Description
'DL'	REAL	none	Output only - the directed distance from (XPOS,YPOS) to the left edge of the text-extent box for the last string, measured in the direction ANGD-180. A negative value means that the left edge is in the direction ANGD from (XPOS,YPOS). If the last string was effectively blank, 'DL' will be -1E6. If 'MA' is zero, 'DL' is a value in the fractional coordinate system; otherwise it is a value in the same system in which the argument XPOS was given.
'DO'	INTEGER	1	The drawing order for the characters. If the value is positive, characters are drawn in the order in which they appear in the input character string; otherwise, they are drawn in the opposite order. If the absolute value is a 1, all shadows are drawn, followed by all character bodies, followed by all outlines. If the absolute value is 2 or greater, each character is drawn completely (including its shadow, its body, and its outline) before moving on to the next character; this can be used, together with 'SS', to “stack” characters so that they apparently overlap one another.
'DR'	REAL	none	Output only - the directed distance from (XPOS,YPOS) to the right edge of the text-extent box for the last string, measured in the direction ANGD. A negative value means that the right edge is in the direction ANGD-180 from (XPOS,YPOS). If the last string was effectively blank, 'DR' will be -1E6. If 'MA' is zero, 'DR' is a value in the fractional coordinate system; otherwise it is a value in the same system in which the argument XPOS was given.
'DT'	REAL	none	Output only - the directed distance from (XPOS,YPOS) to the top edge of the text-extent box for the last string, measured in the direction ANGD+90. A negative value means that the bottom edge is in the direction ANGD-90 from (XPOS,YPOS). If the last string was effectively blank, 'DT' will be -1E6. If 'MA' is zero, 'DT' is a value in the fractional coordinate system; otherwise, it is a value in the same system in which the argument YPOS was given.
'FB’	REAL	.00003	Fidelity parameter for Bezier curves used to draw the filled fonts. The value specifies how close the interpolated curve should be to the actual Bezier curve and is specified as a fraction of the height of the plotter frame. This is actually not an internal parameter of PLOTCHAR, but of a Bezier-curve package written by Fred Clare. When you tell PLOTCHAR to reset the value of 'FB', it just passes the given value along to the routine BZSETR as the value of the internal parameter 'FTL'.
'FC'	CHAR	':' (a colon)	Function-code character.

TABLE 1. Internal Parameters of PLCHHQ.

Name	Type	Default	Description
'FN'	CHAR or INTEGER	0	Font number. The default value, 0, implies the use of the PWRITX database. Use a positive value “n” to force the use of fontcap database “n”. Use a character string, as shown in table 2, below, in a call to PCSETC; a call to PCGETC will return one of these character strings.
'IH'	REAL	13.	Digitized indexical height.
'IS'	REAL	7.	Offset to be used when subscripting or superscripting a character of indexical size, in digitization units.
'IV'	REAL	20.	Digitized indexical vertical spacing.
'IW'	REAL	12.	Digitized indexical width.
'MA'	INTEGER	0	The mapping flag. The value zero says that no mapping is to occur; non-zero values say that PCMPXY is to be called and select particular mappings.
'OC'	INTEGER	1	The outline color specifier. The value “-1” says that outline color is unspecified; a value of 0 or greater selects a particular color index.
'OF'	INTEGER	0	The outline flag. The value 0 says that outlines are not to be drawn; a non-zero value says that outlines are to be drawn.
'OL'	REAL	0.	The outline line width. A value less than or equal to zero says that outline width is unspecified; a value greater than zero is the desired line width, as a fraction of “normal”.
'OR'	REAL	0.	The out-of-range flag. The value 0. says that no values returned by the routine PCMPXY are “out of range”; any other value is the value to be used as an out-of-range signal, indicating that a point to be mapped is invisible under the current mapping.
'PH'	REAL	21.	Digitized principal height.
'PS'	REAL	10.	Offset to be used when subscripting or superscripting a character of principal size, in digitization units.
'PV'	REAL	32.	Digitized principal vertical spacing.
'PW'	REAL	16.	Digitized principal width.
'QU'	INTEGER	0	Quality flag. The value 0 means to use the high-quality characters, 1 means to use the medium-quality characters of PLCHMQ, and 2 means to use the “low-quality” characters of PLCHLQ.
'SA'	REAL	.88888888...	A multiplier for the sizes of all characters written directly by PLCHHQ, introduced in version 3.2 of NCAR Graphics in order to make characters written by PLCHHQ the same size as those written by PLCHMQ. The default value is (16/21)(7/6). Those users who want PLCHHQ to produce the same size characters that it did before may set 'SA' to 1., but this is not recommended.
'SC'	INTEGER	0	The shadow color specifier. The value “-1” says that shadow color is unspecified; a value of 0 or greater selects a particular color index.

TABLE 1. Internal Parameters of PLCHHQ.

Name	Type	Default	Description
'SF'	INTEGER	0	The shadow flag. The value 0 says that shadows are not to be drawn; a non-zero value says that shadows are to be drawn.
'SL'	REAL	0.	The shadow line width. A value less than or equal to zero says that shadow line width is unspecified; a value greater than zero is the desired line width, as a fraction of “normal”.
'SS'	REAL	0.	Subtract-space flag. When this parameter is non-zero, it reduces the space between characters. If a negative value is used, it is interpreted in digitization units. If a positive value is used, it is interpreted as a multiplier of the nominal character width specified by the argument SIZE (and, in the case of the PWRITX database, one of 'PW', 'IW', or 'CW', whichever is appropriate).
'SX'	REAL	-.05	The X offset for character shadows, stated as a fraction of principal character height. This is actually an offset in the direction ANGD.
'SY'	REAL	-.05	The Y offset for character shadows, stated as a fraction of principal character height. This is actually an offset in the direction ANGD+90.
'TE'	INTEGER	0	The text-extent computation flag. Zero means do not compute text-extent quantities, non-zero means do compute them. If 'TE' is non-zero and the value of the argument ANGD, in a call to PLCHHQ, is exactly 360. instead of 0., no characters are drawn, but the text-extent quantities are still computed. This provides a way to get these quantities prior to actually drawing a particular string.When 'MA' is zero (mapping is turned off), values are returned in the fractional coordinate system; otherwise, they are in the same coordinate system as that in which the arguments XPOS and YPOS are given, prior to the mapping implied by the value of 'MA'.
'UN'	INTEGER	3	A FORTRAN logical unit number to be used in reading the data that defines the various character sets. On Unix systems, all such data are read using C I/O, so the value of 'UN' is ignored; if you retrieve its value on such a system, you may get something a mite strange.
'XB'	REAL	0.	For retrieval only. The X position at the beginning of the last string written by PLCHHQ, in the fractional system if 'MA' is zero, and in the same system as XPOS and YPOS, otherwise.
'XC'	REAL	0.	For retrieval only. The X position at the center of the last character written by PLCHHQ, in the fractional system if 'MA' is zero, and in the same system as XPOS and YPOS, otherwise.
'XE'	REAL	0.	For retrieval only. The X position at the end of the last string written by PLCHHQ, in the fractional system if 'MA' is zero, and in the same system as XPOS and YPOS, otherwise.

TABLE 1. Internal Parameters of PLCHHQ.

Name	Type	Default	Description
'YB'	REAL	0.	For retrieval only. The Y position at the beginning of the last string written by PLCHHQ, in the fractional system if 'MA' is zero, and in the same system as XPOS and YPOS, otherwise.
'YC'	REAL	0.	For retrieval only. The Y position at the center of the last character written by PLCHHQ, in the fractional system if 'MA' is zero, and in the same system as XPOS and YPOS, otherwise.
'YE'	REAL	0.	For retrieval only. The Y position at the end of the last string written by PLCHHQ, in the fractional system if 'MA' is zero, and in the same system as XPOS and YPOS, otherwise.
'ZX'	REAL	1.	A "zoom factor" for character width. The default value is 1. Setting 'ZX' to a value other than 1 is equivalent to prepending an "X" function code to each input character string. For example, setting 'ZX' = 1.5 is equivalent to prepending ":X150:" to each input character string. If both 'ZX' and 'ZZ' are set, the product of the two values is used as a multiplier for the width of characters.
'ZY'	REAL	1.	A "zoom factor" for character height. The default value is 1. Setting 'ZY to a value other than 1 is equivalent to prepending a "Y" function code to each input character string. For example, setting 'ZY' = .75 is equivalent to prepending ":Y75:" to each input character string. If both 'ZY' and 'ZZ' are set, the product of the two values is used as a multilplier for the height of characters.
'ZZ'	REAL	1.	A "zoom factor" for character width and height. The default value is 1. Setting 'ZZ' to a value other than 1 is equivalent to prepending a "Z" function code to each input character string. For example, setting 'ZZ' = 2.5 is equivalent to prepending ":Z250:" to each input character string. If both 'ZX' and 'ZZ' are set, the product of the two values is used as a multiplier for the width of characters; similarly, if both 'ZY' and 'ZZ' are set, the product of the two values is used as a multiplier for the height of characters.

The following table relates font numbers and font names, either of which may now be used in setting the value of the internal parameter 'FN':

TABLE 2. Font Numbers vs. Font Names

Font No.	Font Name	Type of Font
0	PWRITX DATABASE	Stroked fonts.
1	DEFAULT	Stroked font defined by fontcap.
2	CARTOGRAPHIC_ROMAN	Stroked font defined by fontcap.
3	CARTOGRAPHIC_GREEK	Stroked font defined by fontcap.
4	SIMPLEX_ROMAN	Stroked font defined by fontcap.
5	SIMPLEX_GREEK	Stroked font defined by fontcap.

TABLE 2. Font Numbers vs. Font Names

Font No.	Font Name	Type of Font
6	SIMPLEX_SCRIPT	Stroked font defined by fontcap.
7	COMPLEX_ROMAN	Stroked font defined by fontcap.
8	COMPLEX_GREEK	Stroked font defined by fontcap.
9	COMPLEX_SCRIPT	Stroked font defined by fontcap.
10	COMPLEX_ITALIC	Stroked font defined by fontcap.
11	COMPLEX_CYRILLIC	Stroked font defined by fontcap.
12	DUPLEX_ROMAN	Stroked font defined by fontcap.
13	TRIPLEX_ROMAN	Stroked font defined by fontcap.
14	TRIPLEX_ITALIC	Stroked font defined by fontcap.
15	GOTHIC_GERMAN	Stroked font defined by fontcap.
16	GOTHIC_ENGLISH	Stroked font defined by fontcap.
17	GOTHIC_ITALIAN	Stroked font defined by fontcap.
18	MATH_SYMBOLS	Stroked font defined by fontcap.
19	SYMBOL_SET1	Stroked font defined by fontcap.
20	SYMBOL_SET2	Stroked font defined by fontcap.
21	HELVETICA	Filled font defined by fontcap.
22	HELVETICA-BOLD	Filled font defined by fontcap.
25	TIMES-ROMAN	Filled font defined by fontcap.
26	TIMES-BOLD	Filled font defined by fontcap.
29	COURIER	Filled font defined by fontcap.
30	COURIER-BOLD	Filled font defined by fontcap.
33	GREEK	Filled font defined by fontcap.
34	MATH-SYMBOLS	Filled font defined by fontcap.
35	TEXT-SYMBOLS	Filled font defined by fontcap.
36	WEATHER1	Filled font defined by fontcap.
37	WEATHER2	Filled font defined by fontcap.
121	O_HELVETICA	Outline font for filled font 21.
122	O_HELVETICA-BOLD	Outline font for filled font 22.
125	O_TIMES-ROMAN	Outline font for filled font 25.
126	O_TIMES-BOLD	Outline font for filled font 26.
129	O_COURIER	Outline font for filled font 29.
130	O_COURIER-BOLD	Outline font for filled font 30.
133	O_GREEK	Outline font for filled font 33.
134	O_MATH-SYMBOLS	Outline font for filled font 34.
135	O_TEXT-SYMBOLS	Outline font for filled font 35.
136	O_WEATHER1	Outline font for filled font 36.
137	O_WEATHER2	Outline font for filled font 37.

The plots produced by the NCAR Graphics example named “epltch” include samples of all characters from all of the above fonts.

3.12 History

Originally implemented to make the Hershey character sets more accessible, with the name PWRX.

Made portable in 1978 and given the name PWRITX.

Made to be FORTRAN 77 and GKS compatible, in August, 1984.

Massively rewritten in February-March, 1988, and renamed PLCHHQ.

Overhauled in June, 1990, principally to provide access to fontcap databases.

Modified in November and December, 1992, to provide access to filled fonts, to implement shadows, outlines, and boxes, and to implement mapping of characters.

4.0 PLCHMQ (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)

PLCHMQ draws characters of “medium quality”. It does this by drawing lines, just as PLCHHQ does, but it does not produce quite such fancy characters. No function codes may be used. Using PLCHMQ to draw a given string of characters will create a larger metafile than if PLCHLQ were used, which may be a disadvantage. However, it may also be more dependable, in that it does not depend on capabilities the translator may or may not have.

Note: The name PCMEQU, which better fits the naming pattern used for other routines in the package, may be used as a synonym for PLCHMQ.

4.1 Usage

PLCHMQ is called using either of the following FORTRAN statements:

```
CALL PLCHMQ (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
CALL PCMEQU (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
```

4.2 Arguments

XPOS and **YPOS** are input values of type REAL, specifying positioning coordinates for the characters to be drawn. If the internal parameter 'MA' has the value 0, these are given in the current user coordinate system; otherwise, they are given in an arbitrary X/Y coordinate system, as implied by the value of 'MA' and the nature of the routine CPMPXY. (For example, if 'MA' has the value 1 and the default version of CPMPXY is being used, then XPOS is a longitude, in degrees, and YPOS is a latitude, in degrees.) The argument CNTR (described below) specifies how the characters are to be positioned relative to the point (XPOS,YPOS).

CHRS is of type CHARACTER. It contains a character string to be drawn. The number of characters in CHRS is taken to be LEN(CHRS); to use characters "m" through "n" from a character variable CHRS, use the FORTRAN-77 substring notation "CHRS(m:n)". The string may include any of the 95 characters space, exclamation point, double quote, pound sign, dollar sign, percent sign, ampersand, apostrophe, left parenthesis, right parenthesis, asterisk, plus sign, comma, minus sign, period, slash, 0-9, colon, semi-colon, less than sign, equals sign, greater than sign, question mark, at sign, A-Z, left square bracket, backslash, right square bracket, hat, underscore, backwards quote, a-z, left curly bracket, vertical bar, right curly bracket, and tilde. Function codes, like those in calls to PLCHHQ, may not be used.

SIZE specifies the desired character size. If the internal parameter 'MA' is zero, then the following comments apply:

- If SIZE is less than or equal to zero, its absolute value specifies the size as a multiple of a default digitized size on a 1024x1024 grid, on which blanks are 16 units wide.
- If SIZE is greater than zero, but less than one, it specifies the desired width of a blank as a fraction of the distance across the plotter frame. This is the recommended scheme.
- If SIZE is greater than or equal to one, it specifies the desired width of a blank in plotter coordinates, as defined by default or by a user's call to the SPPS routine SETI. Note that use of the routine SETI is now discouraged.

If 'MA' is non-zero, then SIZE is the desired width of a blank as a value in the X/Y coordinate system in which XPOS and YPOS are given.

Capital letters are 4/6 as wide as a blank and, by default, 7/4 as high as they are wide (the internal parameter 'HW' may be used to change the latter value). The distance from the center of each character to the center of the next is constant and is equal to one blank width.

Note that, if the desired character height ("h") is known, one should use SIZE = (6/7) * h.

SIZE is defined in such a way as to be consistent with PLCHHQ. If one changes a "CALL PLCHHQ" to a "CALL PLCHMQ" (assuming that no function codes are used in the string, of course), the results will be much the same, but the length of the string will be different. If one uses SIZE less than zero and one has reset the PLCHHQ internal parameter 'PW' to something other than 16, the difference in the output of the two routines will be greater.

ANGD is the angle, in degrees counterclockwise from the positive X axis, at which the character string is to be written.

CNTR is the centering option, as follows:

- CNTR = -1. means that (XPOS,YPOS) is the center of the left edge of the first character.
- CNTR = 1. means that (XPOS,YPOS) is the center of the right edge of the last character.
- CNTR = s means that (XPOS,YPOS) is a point obtained by linear interpolation along the line joining the two points mentioned above (the first point being associated with the value -1 and the second with the value +1). The value “0.” gives the midpoint of the line.

Upon return from PLCHMQ, all arguments are unchanged.

4.3 Internal Parameters

The internal parameter 'HW' specifies the desired ratio of the character height (the height of a capital) to the character width (excluding white space). The default value of 'HW' is 1.75, reflecting the fact that the capitals are digitized to be 7 units high and 4 units wide and 1.75 is therefore their natural aspect ratio.Negative values of 'HW' may be used. The absolute value will be used as the ratio, but, in addition, PLCHHQ will be prohibited from changing 'HW' (which it otherwise does when the quality flag 'QU' is set to 1).

PLCHMQ reacts properly to non-zero values of 'MA' and 'OR', which are used to request mapping of characters through the routine PCMPXY, as described for PLCHHQ, above. Characters that are partly visible and partly invisible are clipped at the visible/invisible boundary.

4.4 History

Implemented for use by DASHCHAR with the name PWRY.

Made portable in January, 1977, and given the name PWRITY.

Converted to FORTRAN 77 and GKS in July, 1984.

Massively rewritten in February-March, 1988, and renamed PLCHMQ. Many ASCII characters were added to the dataset at that time.

Modified in November, 1992, to implement character mapping and clipping.

5.0 PLCHLQ (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)

PLCHLQ draws characters of “low quality” by calling the GKS character-drawing routines. No function codes may be used. Using PLCHLQ to draw a given string of characters will create a smaller metafile than if PLCHHQ or PLCHMQ were used; the results will depend on capabilities of the translator.

Note: The name PCLOQU, which better fits the naming pattern used for other routines in the package, may be used as a synonym for PLCHLQ.

5.1 Usage

PLCHLQ is called using either of the following FORTRAN statements:

```
CALL PLCHLQ (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
CALL PCLOQU (XPOS,YPOS,CHRS,SIZE,ANGD,CNTR)
```

5.2 Arguments

XPOS and **YPOS** are input values of type REAL, specifying positioning coordinates for the characters to be drawn. If the internal parameter 'MA' has the value 0, these are given in the current user coordinate system; otherwise, they are given in an arbitrary X/Y coordinate system, as implied by the value of 'MA' and the nature of the routine CPMPXY. (For example, if 'MA' has the value 1 and the default version of CPMPXY is being used, then XPOS is a longitude, in degrees, and YPOS is a latitude, in degrees.) The argument CNTR (described below) specifies how the characters are to be positioned relative to the point (XPOS,YPOS).

CHRS is of type CHARACTER. It contains a character string to be drawn. The number of characters in CHRS is taken to be LEN(CHRS); to use characters “m” through “n” from a character variable CHRS, use the FORTRAN-77 substring notation “CHRS(m:n)”. The string may include any of the 95 characters space, exclamation point, double quote, pound sign, dollar sign, percent sign, ampersand, apostrophe, left parenthesis, right parenthesis, asterisk, plus sign, comma, minus sign, period, slash, 0-9, colon, semi-colon, less than sign, equals sign, greater than sign, question mark, at sign, A-Z, left square bracket, backslash, right square bracket, hat, underscore, backwards quote, a-z, left curly bracket, vertical bar, right curly bracket, and tilde. Function codes, like those in calls to PLCHHQ, may not be used.

SIZE specifies the desired character size. If the internal parameter 'MA' is zero, then the following comments apply:

- If SIZE is less than or equal to zero, its absolute value specifies the size as a multiple of a default digitized size on a 1024x1024 grid, on which blanks are 16 units wide.
- If SIZE is greater than zero, but less than one, it specifies the desired width of a blank as a fraction of the distance across the plotter frame. This is the recommended scheme.
- If SIZE is greater than or equal to one, it specifies the desired width of a blank in plotter coordinates, as defined by default or by a user’s call to the SPPS routine SETI. Note that use of the routine SETI is now discouraged.

If 'MA' is non-zero, then SIZE is the desired width of a blank as a value in the X/Y coordinate system in which XPOS and YPOS are given.

Note that SIZE is defined in such a way as to be consistent with PLCHHQ.

ANGD is the angle, in degrees counterclockwise from the positive X axis, at which the character string is to be written.

CNTR is the centering option, as follows:

- CNTR < 0. means that (XPOS,YPOS) is the center of the left edge of the first character.
- CNTR > 0. means that (XPOS,YPOS) is the center of the right edge of the last character.
- CNTR = 0. means that (XPOS,YPOS) is the midpoint of the string.

Note that this argument is not quite the same as that for PLCHHQ (though it does include the three most useful cases).

Upon return from PLCHLQ, all arguments are unchanged.

5.3 Internal Parameters

PLCHLQ reacts to non-zero values of 'MA' and 'OR', which are used to request mapping of characters through the routine PCMPXY, as described for PLCHHQ, above. However, it doesn't react in quite the same way: Mapping will affect the position of the string written by a call to PLCHLQ, the angle at which it is written, and the size of the characters used. The shape of the characters will not be affected (and cannot be, since the characters are drawn by calling the GKS routine GTX instead of GPL). If the point (XPOS,YPOS) is visible under the mapping, the whole string is considered to be visible; otherwise, the whole string is considered to be invisible. The results can be pretty crude; for that reason, mapping is not recommended.

5.4 History

Written in March, 1988, as the third member of the PLCHxQ family. Based on WTSTR, from SPPS.

Modified in November, 1992, to implement character mapping and clipping.

6.0 PCGETx (PNAM,xVAL)

These routines are used to get the values of internal parameters.

6.1 Usage

Use one of the FORTRAN statements

```
CALL PCGETC (PNAM,CVAL)
CALL PCGETI (PNAM,IVAL)
CALL PCGETR (PNAM,RVAL)
```

Note that the last character of the subroutine name matches the type of the second argument (“C” implies an argument of type “CHARACTER”, “I” an argument of type “INTEGER”, and “R” an argument of type “REAL”).

6.2 Arguments

PNAM is a character string specifying the name of the parameter to get. The name must appear as the first two characters of the string. If the internal parameter is one of the two ('BC' and 'CC') that are arrays, the index of the desired array element may appear, enclosed in parentheses, in columns 3 and following. Other characters may be used to document the use of the parameter being retrieved; for example, instead of just 'MA', one can use 'MA - MAPPING FLAG' and, instead of 'CC(10)', one can use 'CC(10) - SPECIAL COLOR 10'.

CVAL, **IVAL**, or **RVAL** is a variable to receive the value of the parameter specified by PNAM - of type CHARACTER, INTEGER, or REAL, respectively.

Upon return from one of these routines, the argument PNAM is unchanged and the desired value has been placed in the argument CVAL, IVAL, or RVAL.

7.0 PCSETx (PNAM,xVAL)

These routines are used to set the values of internal parameters.

7.1 Usage

Use one of the FORTRAN statements

```
CALL PCSETC (PNAM,CVAL)
CALL PCSETI (PNAM,IVAL)
CALL PCSETR (PNAM,RVAL)
```

Note that the last character of the subroutine name matches the type of the second argument (“C” implies an argument of type “CHARACTER”, “I” an argument of type “INTEGER”, and “R” an argument of type “REAL”).

7.2 Arguments

PNAM is a character string specifying the name of the parameter to be set. The name must appear as the first two characters of the string. If the internal parameter is one of the two ('BC' and 'CC') that are arrays, the index of the desired array element may appear, enclosed in parentheses, in columns 3 and following. Other characters may be used to document the use of the parameter being retrieved; for example, instead of just 'MA', one can use 'MA - MAPPING FLAG' and, instead of 'CC(10)', one can use 'CC(10) - SPECIAL COLOR 10'.

CVAL, **IVAL**, or **RVAL** is the value to be assigned to the parameter specified by PNAM - of type CHARACTER, INTEGER, or REAL, respectively.

Upon return from one of these routines, all arguments are unchanged.

8.0 PCDLSC (IFCI)

8.1 Usage

Use the FORTRAN statement

```
CALL PCDLSC (IFCI)
```

to define the default list of “special colors” used by PLCHHQ in drawing certain characters from the filled fonts. Such a call will define color indices IFCI, IFCI+1, IFCI+2, . . . , IFCI+15 and it will set all elements of the internal parameter array 'CC' corresponding to indices 1 through 16. At the moment, though such a call **does** define a set of sixteen colors (ranging from blue to red) and set the elements of 'CC', it’s a bit pointless, since there are no characters for which the special colors are used. In the future, there will be a few such (like the state highway symbol, which is normally drawn in a particular pair of colors); at that time, the routine will be of more use.

8.2 Arguments

IFCI is an integer expression specifying the first of a set of sixteen color indices that may be defined by PCDLSC. Make sure that these color indices are not already used.

9.0 PCPNWI (PNAM,IPAI)

9.1 Usage

PCPNWI is a character function. Its type must be declared, in any routine in which it is used, by inserting the following statement in that routine:

```
CHARACTER*16 PCPNWI
```

Then, the value of the function reference

```
PCPNWI(PNAM,IPAI)
```

is the name of element number IPAI in the internal parameter array whose name is given by the first two characters of the character string PNAM. For example, if IPAI has the value 3, then the value of

```
PCPNWI ('CC - CHARACTER COLORS',IPAI)
```

is the character string 'CC(3)' (padded with blanks to the right) and the call

```
CALL PCSETI (PCPNWI('CC - CHARACTER COLORS',IPAI),62)
```

will therefore give element 3 of the internal parameter array 'CC' the value 62. (In this particular case, of course, it would be easier to write

```
CALL PCSETI ('CC(3) - CHARACTER COLOR 3',62)
```

but that misses the point that the value of the variable IPAI will not always be known.)

9.2 Arguments

PNAM is a character string, the first two characters of which are the name of an internal parameter array of PLOTCHAR (currently, 'BC' or 'CC').

IPAI is an integer expression specifying the index of the element of the internal parameter array of which one wants the name.

10.0 PCRSET

10.1 Usage

This routine may be called to return the values of all of PLOTCHAR’s internal parameters to their default values; it has no arguments.

The internal parameter 'FB' is unaffected by calling PCRSET. To return 'FB' to its default value, use the statement

```
CALL BZSETR (‘FTL',.00003)
```

10.2 Arguments

None.

11.0 PCMPXY (IMAP,XINP,YINP,XOTP,YOTP)

11.1 Usage

This routine is normally not called directly by the user (though it can be). It is called by PLCHHQ, by PLCHMQ, and by PLCHLQ when the user has set the mapping flag 'MA' non-zero to request mapping of characters from one X/Y coordinate system to another. A call to PCMPXY has the following form:

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

The default version of PCMPXY does several useful mappings; a user version may be supplied to do others.

11.2 Arguments

IMAP is an input expression of type INTEGER, specifying which mapping is to be done. Whenever PLOTCHAR routines call PCMPXY, IMAP = 'MA'. The default version of PCMPXY recognizes the following values of IMAP:

IMAP	Type of mapping	XINP	YINP
0	None. When IMAP is zero, the routine PCMPXY is being asked for information about its capabilities with respect to a particular mapping.	Input: the real equivalent of a possible value of IMAP about which information is desired.	Output (real): 0. says that the mapping specified by XINP is not available, 1. that it is available, 2. that its inverse is available, and 3. that both it and its inverse are available.
1	EZMAP projection.	Longitude, in degrees.	Latitude, in degrees.
2	Rho/theta mapping.	Rho, in user units.	Theta, in degrees.
3	Projection from an arbitrary parallelogram in 3-space to an image plane. This is implemented using a package which is currently under development and is not yet generally available.	A multiplier for a unit vector along one edge of the parallelogram.	A multiplier for a unit vector along an intersecting edge of the parallelogram.

IMAP	Type of mapping	XINP	YINP
4	A version of the EZMAP projection that is intended to be used to label specific points on the surface of the globe. See the paragraph entitled “Using 'MA' = 4”, below.	Longitude, in degrees.	Latitude, in degrees.
100	The identity mapping, but returns 1.E12 outside a specified viewport. Used by the utility STITLE to do clipping at the edges of the viewport.	X coordinate, in the fractional system.	Y coordinate, in the fractional system.
other > 0	The identity mapping.	Any real value.	Any real value.
other < 0	An inverse mapping.	A mapped X coordinate.	A mapped Y coordinate.

Note: Values of IMAP less than or equal to zero are not currently used by PLOTCHAR in calling PCMPXY. The scheme described is the same as that used for the routine CPMPXY, in the NCAR Graphics package CONPACK. It is possible that, in the future, values of IMAP less than or equal to zero will occur in calls to PCMPXY, so it is best to be prepared.

XINP and **YINP** are input expressions of type REAL, specifying the X and Y coordinates of a point to be mapped.

Note: When IMAP is less than or equal to zero (which cannot currently happen, but may someday be possible), XINP and YINP are used differently, as shown in the table above.

XOTP and **YOTP** are output variables of type REAL, in which PCMPXY returns the mapped X and Y coordinates of the point. If the point is not visible under the mapping selected (as, for example, when the orthographic projection of EZMAP is being used and the point to be mapped is on the other side of the globe), PCMPXY must return, as a signal that this is the case, whatever special non-zero value has been given to the internal parameter 'OR', so that the calling routine can take appropriate action (to properly clip the character being drawn, for example).

Note: When IMAP is less than or equal to zero (which cannot currently happen, but may someday be possible), XOTP and YOTP are used differently, as shown in the table above.

11.3 Using 'MA' = 4

'MA' = 4 is used to generate labels at specific points on the surface of the globe. When you set 'MA' = 4, then, in every call to PLCHHQ and PLCHMQ, you should use “0.” for each of the arguments XPOS, YPOS, and ANGD. This says that, initially, the label being written is to be placed on the globe at longitude zero and latitude zero and at a zero angle to the equator. Three rotations are then performed to place the label in the desired position: First, it is rotated by an angle PANG about the line joining the center of the globe to the point with latitude zero and longitude zero. Then, it is rotated by an angle PLAT about a line lying in the equatorial plane and perpendicular to the first line. Then, it is rotated by an angle PLON about the polar axis. The net effect of these rotations is to place the label on the globe at the point (PLAT,PLON) in such a way that it makes an angle of PANG with the local eastward-pointing vector. These three rotations cause minimal distortion of the characters. Finally, the EZMAP routine MAPTRA is used to project the label as specified by the current state of EZMAP.

The desired values of PANG, PLAT, and PLON (in degrees) are passed to PCMPXY in the following COMMON block:

```
COMMON /PCMP04/ PANG,PLAT,PLON
```

See the CONPACK example “cpex10”, in which 'MA' = 4 is used to generate labels for the states of the U.S.

12.0 ERROR HANDLING

When a PLOTCHAR 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 PLCHHQ - DATASET NOT LOADED CORRECTLY
```

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 (“PLCHHQ”, 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 PLOTCHAR 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:

```
PCHIQU/PLCHHQ - DATASET NOT LOADED CORRECTLY
```

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

- **PCDLSC - FIRST COLOR INDEX IS LESS THAN ZERO**

  PCDLSC has been called with an argument whose value is less than zero, which is illegal.

- **PCDLSC - UNCLEARED PRIOR ERROR**

  When PCDLSC was called, there was an unrecovered outstanding error. The routine does not continue: it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **PCGETC - UNCLEARED PRIOR ERROR**

  When PCGETC was called, there was an unrecovered outstanding error. The routine does not continue: it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **PCGETC - UNRECOGNIZED PARAMETER NAME**

  The first argument in a call to PCGETC is not recognizable as the name of an internal parameter of EZMAP.

- **PCGETI - UNCLEARED PRIOR ERROR**

  When PCGETI was called, there was an unrecovered outstanding error. The routine does not continue: it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **PCGETR - BOX COLOR ARRAY INDEX IS OUT OF RANGE**

  The parameter name begins with 'BC' and contains a subscript that is outside the range 1 to 3, which is the legal range for that internal parameter.

- **PCGETR - COLOR ARRAY INDEX IS OUT OF RANGE**

  The parameter name begins with 'CC' and contains a subscript that is outside the range from 0 to 16, which is the legal range for that internal parameter.

- **PCGETR - UNCLEARED PRIOR ERROR**

  When PCGETR was called, there was an unrecovered outstanding error. The routine does not continue: it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **PCGETR - UNRECOGNIZED PARAMETER NAME**

The first argument in a call to PCGETR is not recognizable as the name of an internal parameter of EZMAP.

- **PCHIQU - UNCLEARED PRIOR ERROR**

  When PCHIQU was called, there was an unrecovered outstanding error. The routine does not continue: it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **PCLOQU - UNCLEARED PRIOR ERROR**

  When PCLOQU was called, there was an unrecovered outstanding error. The routine does not continue: it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **PCMEQU - UNCLEARED PRIOR ERROR**

  When PCMEQU was called, there was an unrecovered outstanding error. The routine does not continue: it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **PCPNWI - INTERNAL ERROR - SEE CONSULTANT**

  PCPNWI is an internal routine that examines a parameter name string for an embedded subscript. This error message indicates a problem with such a parameter name.

- **PCPNWI - PARAMETER NAME TOO SHORT**

  PCPNWI is an internal routine that examines a parameter name string for an embedded subscript.This error message indicates a problem with such a parameter name.

- **PCSETC - UNCLEARED PRIOR ERROR**

  When PCSETC was called, there was an unrecovered outstanding error. The routine does not continue: it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **PCSETC - UNRECOGNIZED FONT NAME**

  If PCSETC is called with first argument 'FN', the second argument must be the name of a font to be used. The name of the font must be spelled exactly as given in Table 2, “Font Numbers vs. Font Names”, above. In particular, the name must be given in upper case.

- **PCSETC - UNRECOGNIZED PARAMETER NAME**

  The first argument in a call to PCSETC is not recognizable as the name of an internal parameter of EZMAP.

- **PCSETI - UNCLEARED PRIOR ERROR**

  When PCSETI was called, there was an unrecovered outstanding error. The routine does not continue: it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **PCSETR - BOX COLOR ARRAY INDEX IS OUT OF RANGE**

  The parameter name begins with 'BC' and contains a subscript that is outside the range 1 to 3, which is the legal range for that internal parameter.

- **PCSETR - COLOR ARRAY INDEX IS OUT OF RANGE**

  The parameter name begins with 'CC' and contains a subscript that is outside the range from 0 to 16, which is the legal range for that internal parameter.

- **PCSETR - UNCLEARED PRIOR ERROR**

  When PCSETR was called, there was an unrecovered outstanding error. The routine does not continue: it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **PCSETR - UNRECOGNIZED PARAMETER NAME**

  The first argument in a call to PCSETR is not recognizable as the name of an internal parameter of EZMAP.

- **PLCHHQ - DATASET NOT LOADED CORRECTLY**

  Probably indicates an error in the implementation of PLOTCHAR. See a consultant.

- **PLCHHQ - ERROR EXIT FROM GQCNTN**

  One of the GKS “query” routines has returned an error flag. This may mean that GKS is not in the proper state.

- **PLCHHQ - ERROR EXIT FROM GQFACI**

  One of the GKS “query” routines has returned an error flag. This may mean that GKS is not in the proper state.

- **PLCHHQ - ERROR EXIT FROM GQFAIS**

  One of the GKS “query” routines has returned an error flag. This may mean that GKS is not in the proper state.

- **PLCHHQ - ERROR EXIT FROM GQLWSC**

  One of the GKS “query” routines has returned an error flag. This may mean that GKS is not in the proper state.

- **PLCHHQ - ERROR EXIT FROM GQNT**

  One of the GKS “query” routines has returned an error flag. This may mean that GKS is not in the proper state.

- **PLCHHQ - ERROR EXIT FROM GQPLCI**

  One of the GKS “query” routines has returned an error flag. This may mean that GKS is not in the proper state.

- **PLCHHQ - ERROR EXIT FROM GQTXCI**

  One of the GKS “query” routines has returned an error flag. This may mean that GKS is not in the proper state.

- **PLCHHQ - INTERNAL LOGIC ERROR (NCRA TOO BIG) - SEE CONSULT-ANT**

  An “impossible” situation has arisen. Indicates a compiler problem, core clobbering, an error in the implementation or installation of PLOTCHAR, and/or the presence of unfriendly gremlins. See a consultant.

- **PLCHHQ - INTERNAL LOGIC ERROR (NDGU = 0) - SEE CONSULTANT**

  An “impossible” situation has arisen. Indicates a compiler problem, core clobbering, an error in the implementation or installation of PLOTCHAR, and/or the presence of unfriendly gremlins. See a consultant.

- **PLCHHQ - INTERNAL LOGIC ERROR (NPCS TOO BIG) - SEE CONSULTANT**

  An “impossible” situation has arisen. Indicates a compiler problem, core clobbering, an error in the implementation or installation of PLOTCHAR, and/or the presence of unfriendly gremlins. See a consultant.

- **PLCHHQ - INTERNAL LOGIC ERROR (XCP1 OR XCP2 = OORV) - SEE CONSULTANT**

  An “impossible” situation has arisen. Indicates a compiler problem, core clobbering, an error in the implementation or installation of PLOTCHAR, and/or the presence of unfriendly gremlins. See a consultant.

- **PLCHHQ - INTERNAL LOGIC ERROR (XCRA/YCRA TOO SMALL) - SEE CONSULTANT**

  An “impossible” situation has arisen. Indicates a compiler problem, core clobbering, an error in the implementation or installation of PLOTCHAR, and/or the presence of unfriendly gremlins. See a consultant.

- **PLCHHQ - UNCLEARED PRIOR ERROR**

  When PLCHHQ was called, there was an unrecovered outstanding error. The routine does not continue: it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **PLCHLQ - ERROR EXIT FROM GQCHH**

  One of the GKS “query” routines has returned an error flag. This may mean that GKS is not in the proper state.

- **PLCHLQ - ERROR EXIT FROM GQCHUP**

  One of the GKS “query” routines has returned an error flag. This may mean that GKS is not in the proper state.

- **PLCHLQ - ERROR EXIT FROM GQCNTN**

  One of the GKS “query” routines has returned an error flag. This may mean that GKS is not in the proper state.

- **PLCHLQ - ERROR EXIT FROM GQNT**

  One of the GKS “query” routines has returned an error flag. This may mean that GKS is not in the proper state.

- **PLCHLQ - ERROR EXIT FROM GQTXAL**

  One of the GKS “query” routines has returned an error flag. This may mean that GKS is not in the proper state.

- **PLCHLQ - ERROR EXIT FROM GQTXP**

One of the GKS "query" routines has returned an error flag. This may mean that GKS is not in the proper state.

- **PLCHLQ - UNCLEARED PRIOR ERROR**

  When PLCHLQ was called, there was an unrecovered outstanding error. The routine does not continue: it forces the error message for the outstanding error to be printed and then substitutes this one for it.

- **PLCHMQ - UNCLEARED PRIOR ERROR**

  When PLCHMQ was called, there was an unrecovered outstanding error. The routine does not continue: it forces the error message for the outstanding error to be printed and then substitutes this one for it.

13.0 EXAMPLES

Three NCAR Graphics examples are recommended to illustrate the use of PLOTCHAR; they are named "tpltch", "epltch", and "cpex10"; on a Unix system, the code for these examples may be obtained and run with the commands

```
ncargex tpltch
ncargex epltch
ncargex cpex10
```

The first of these commands will produce a metafile called "tpltch.ncgm", with a single frame demonstrating minimal functioning of the package.

The second command will produce a metafile called "epltch.ncgm" containing twelve plots on six frames. These plots are as follows:

- The first four plots appear on a single frame; together, they show the complete complex character set.The next four plots also appear on a single frame; together, they show the complete duplex character set. Each plot shows a rectangular grid of characters. The character in a given column and row is the character that would be written by PLCHHQ using the function code shown at the top of the column with the standard FORTRAN character shown at the beginning of the row. (Alternatively, each character could be selected by using as a function code the octal number obtained by summing the four-digit octal number for the column and the two-digit octal number for the row.)

  Note: If putting four plots on each of the first two frames makes the characters too small, one can change the data statement defining the variable ICMP, in the test program, to make it a 0, which will cause each of the first eight plots to be put on a frame by itself.

- The ninth plot, on frame 3, shows various capabilities of PLCHHQ.

- The tenth plot, on frame 4, shows the entire medium-quality character set for five different values of the height-to-width ratio.

- The eleventh plot, on frame 5, shows all of the characters from the fontcap-defined databases to which access was added in June of 1990. This plot can be used to select characters from these databases (using the fact that the characters in font 1 are just the ASCII characters). For example, if you want the character from font 11 that looks like an X, you should use the input string ':F11:U'. Similarly, to use the treble clef from font 20, use the input string ':F20:T'.
- The twelfth plot, on frame 6, illustrates various capabilities that were added in June of 1990.
- The thirteenth plot, on frame 7, shows all of the filled fonts that were added in November of 1992 and may be used like the eleventh plot to select characters from these fonts.
- The fourteenth plot, on frame 8, shows various capabilities that were added to the package in November and December of 1992. A color device is necessary to view this plot.

The third command will produce a metafile called “cpex10.ncgm”, containing 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 along with it.