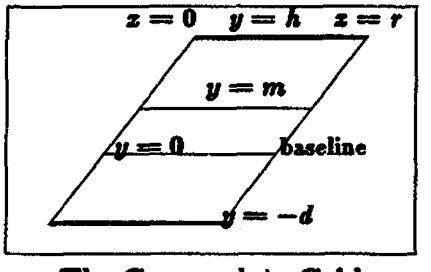
Computer Calligraphy

Georgia K. M. Tobin
Office of Research, OCLC Online Computer Library Center, Inc.

CuPl is a font I designed using Thom Hickey's Pascal version of METAFONT on an Apollo microcomputer. It is intended to capture the flavor of a type of calligraphy called Copperplate, a form of script originally engraved directly onto smooth copper plates using a stylus. This brief report will provide a general description of the design and implementation in METAFONT of *CuPl*, focusing its attention on the set-up of the pens and subroutines used in drawing the characters. In designing *CuPl*, I followed the specifications for the letter forms set down in, "Calligraphy in the Copperplate Style," by Herb Kaufman and Geri Homelsky. According to them, Copperplate letters are characterized by thick downward strokes and hairline-thin upward strokes, and the characters' axes have a uniform slant of 54 degrees from the baseline.

Cubase is the file that fills the same function for *CuPl* that *cmbase* fills for *cmr*, i.e., it contains all the assorted definitions and subroutines that take care of the nasty details that METAFONT needs to know. *Cubase* began life as a clone of the *base* that Thom Hickey used in designing Chel (Computer Helvetica), and only gradually developed its own character. The first and most obvious change was to adjust the grid upon which characters are designed to account for the slant and proportions of Copperplate-style characters. The first requirement is easily satisfied by setting *trxy* (one of METAFONT's transformation parameters) to 0.75; this gives the designated slant of 54 degrees from the baseline. (Of course, other *trxy*s may be used. While these do not give the canonical Copperplate look, they result in some interesting and useful fonts. For instance, I used a *trxy* of 0 to produce a non-slanted variant called Tinplate (*TnPl*) which can be used on the Apollo video display.)

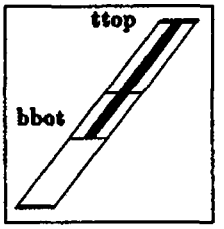
The second requirement entails setting up the proper heights for upper case characters (represented by the value of h), and lower case characters (represented by the value of m) and the proper descender depth (represented by the value of d). Kaufman and Homelsky specify that the baseline is three units above the descender; the tops of lower case letters extend two units above the baseline; and the tops of upper case characters extend three units above the tops of lower case letters. Total design size, then, equals eight units; therefore, h must equal 5/8 of design size, m must equal 2/5 of h, and d must equal 3/8 of design size to get the grid we want, namely:



The Copperplate Grid

Next, I defined a series of horizontal and vertical pens finer than those required by Chel. The horizontal pens are called w_1, w_2, w_3, w_4, w_5, w_6, and w_7; the vertical pens are w_{101}, w_{103}, and w_{105}. w_7 is the thickest pen required by *CuPl* and is used in most of the downward strokes of upper case characters. It gives a stroke of about 4.5 points when drawing a character with a design size of 80 points. w_5 is the pen *CuPl* used for most of the downward strokes in lower case characters; it produces a stroke of about 3.5 points when drawing a character with a design size of 80 points. w_1 and w_{101} are hairline-thin "slow grows"; that is, pens whose widths grow relatively slowly as boldness and/or expansion increases. These pens draw strokes wider than one pixel only at the greatest boldness and/or expansion.

A good deal of a font's character depends upon the subroutines with which it is drawn. In developing the *CuPt* subroutines, I took into account the fact that, according to Kaufman and Homelsky, there are nine basic strokes which occur in almost all the lower case letters and some of the upper case letters. Not all of these basic strokes became subroutines, and some subroutines were required for common shapes not described by those strokes; but all subroutines *are* described in fairly tiresome detail below.

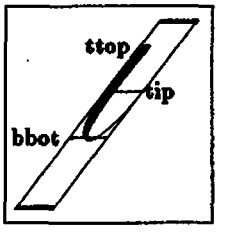


Basic Stroke One

Basic Stroke One is simply a straight line, but, because one or both of its ends may or may not taper, it is best produced by a call to either *Strone* or *Ucstrone*. *Strone* is used for lower case letters, i.e., it uses w_5 for most of the length of the stroke. It requires four parameters: the top point on the stroke (*ttop*), the bottom point on the stroke (*bbot*), the index of the pen used at the top and the index of the pen used at the bottom. These last two are needed to control taper at the ends. For example, the call

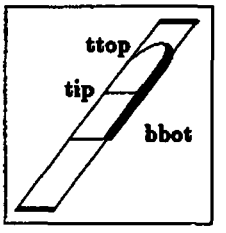
call *strone*(1, 2, 3, 3);

produces a vertical line from 1 to 2 which starts with a w_3, gradually widens to a w_5, and then tapers down again to a w_3. *Ucstrone* is used for upper case letters, i.e. it uses a w_7 for most of the stroke. It only requires two parameters, the top point and the bottom point, because tapering of the stroke from a w_5 is done automatically at both ends.



Basic Stroke Two

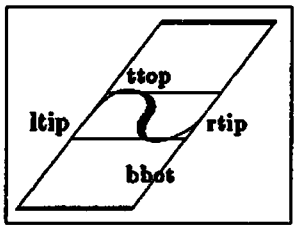
Basic Stroke Two can be produced by defining the coordinates of the topmost point on the stroke (*ttop*), the lowest point the rounded bottom reaches (*bbot*), the ending point on the stroke (*tip*), and the index of the pen used at the top (w_3 for tapering lower case letters, w_5 for non-tapering lower case letters or tapering upper case letters, w_7 for non-tapering upper case letters), and then passing those parameters to either *Strtwo* (for lower case letters) or *Ucstrtwo* (for upper case letters).



Basic Stroke Three

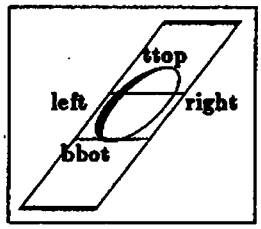
Basic Stroke Three is the reverse of Basic Stroke Two. It can be produced with a call to either *Strthree* or *Ucstrthree* for either lower or upper case, respectively. *Strthree* takes three parameters: the bottom point on

the straight stem of the stroke (*bbot*), the highest point the rounded top reaches (*ttop*), and the ending point on the stroke (*tip*). *Ucstrthree* requires those three parameters, as well as the index of the pen used at the bottom of the straight stem; w_5 is used if the stroke is to taper at the bottom, w_7 if it does not.



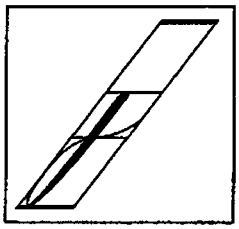
Basic Stroke Four

Basic Stroke Four is a combination of strokes two and three, and may be produced with a call to *Strfour* for lower case; this stroke does not occur in any upper case letters. *Strfour* requires four parameters: the highest point of the rounded top (*ttop*), the lowest point of the rounded bottom (*bbot*), the leftmost point where the stroke begins (*ltip*), and the rightmost point where it ends (*rtip*). All tapering required is handled by the subroutine (which is a nice way of saying that you have no control over the pens used.)

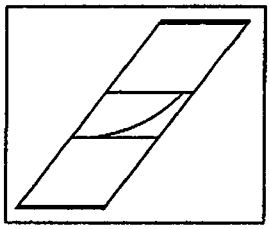


Basic Stroke Five

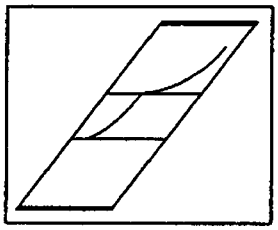
Basic Stroke Five is an oval which is thick on the left hand side and thin on the right. This stroke does not occur in any upper case letter. It may be produced by calling *Strfive*. The parameters required are: the highest point (*ttop*), the lowest point (*bbot*), the rightmost point (*right*), and the leftmost point (*left*). All tapering is handled by the subroutine.



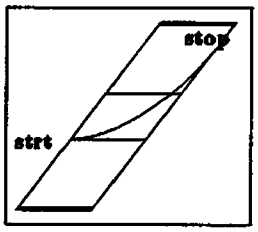
Basic Stroke Six



Basic Stroke Seven

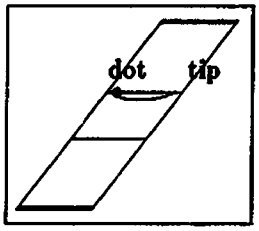


Basic Stroke Eight

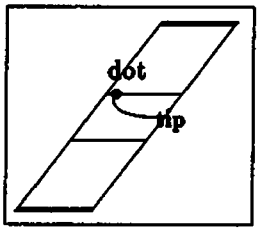


Hairarc

Basic Stroke Six does not occur often enough to merit its own subroutine, and is produced on an *ad hoc* basis. Basic Strokes Seven and Eight are adequately handled by, respectively, one or two calls to the subroutine *Hairarc*, which produces a concave arc from the first point it receives (*strt*) to the second point it receives (*stop*) using a hairline thin pen.



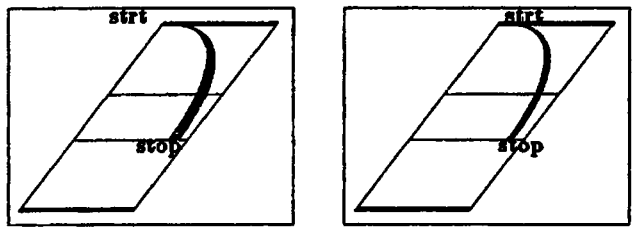
Basic Stroke Nine



Dotloop

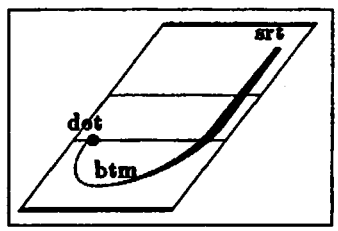
Basic Stroke Nine is produced by a call to *Strnine*. This subroutine requires two parameters: the point at which the dot appears (*dot*), and the ending point of the hairline tail (*tip*). A similar stroke is produced by *Dotloop*, which takes the same two parameters, but produces a tail with a shallower curve.

In addition to the basic strokes that Kaufman and Homelsky describe, there are a number of strokes which occur more or less frequently in Copperplate letters which are also handled by subroutines in *cubase*. These are described below.



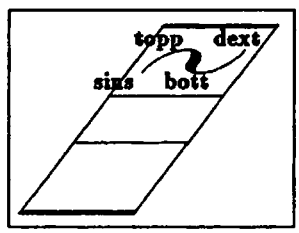
Arc **Growarc**

Arc is used to produce a tapering arc. It requires three parameters: the thinnest point on the arc (drawn with a w_1) (*strt*), the thickest point on the arc (*stop*), and the index of the pen used at the thickest point. *Growarc* also produces an arc given two points, *strt* and *stop*, but allows the user to specify both the pen with which we are to start drawing at *strt* and the pen with which we finish drawing at *stop*.



Dotstroke

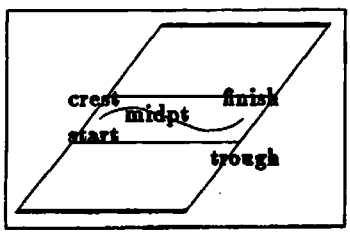
A curving stroke, which tapers at one end and concludes with a dot at the other, is produced by the subroutine *Dotstroke*. *Dotstroke* requires three parameters: the highest and rightmost point on the stroke (*srt*), the nadir of the stroke's rounded bottom (*btm*), and the end point where the dot appears (*dot*).



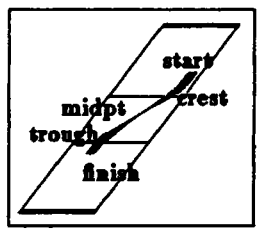
Flounce

This decorative stroke, which appears in several upper case letters, is produced by a call to *Flounce*. Four parameters are required: the leftmost point (*sins*), the rightmost point (*dext*), the highest point (*topp*) and the lowest point (*bott*). All required tapering is handled automatically within the subroutine.

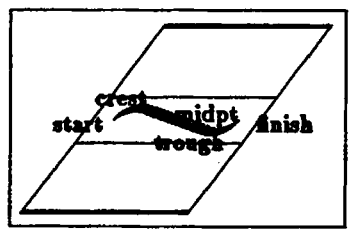
Several subroutines are used to produce smooth, symmetrical waves. Those which have a crest followed by a trough are produced by *Zhair*, *Zsquigl*, or *Qsquigl*.



Zhair

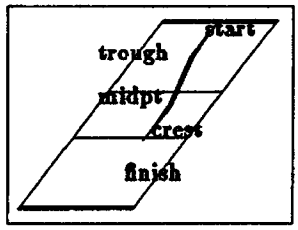


Qsquigl

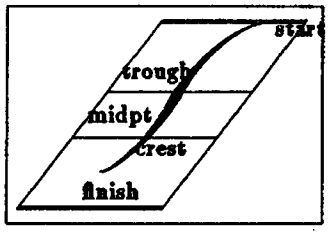


Zsquigl

Zsquigl, *Qsquigl* and *Zhair* all take the following parameters: the wave's starting point (*start*), the y-value of the turning point at the top of the crest (*crest*), the mid-point of the wave (*midpt*), the y-value of the turning point at the bottom of the crest (*trough*), the wave's ending point (*finish*), and a variable equal to the reciprocal of the ending slope. *Zhair* draws the entire wave thus defined with a hairline thin cpen, and so does not require any pen width parameters. *Qsquigl* and *Zsquigl* are passed two pen width parameters before the slope variable is given: the index of the pen with which we start drawing and the index of the pen with which we conclude. *Qsquigl* differs from *Zsquigl* in that *Zsquigl* is designed for waves with a horizontal orientation (i.e., it draws with a vpen), whereas *Qsquigl* is designed for waves with a vertical orientation (i.e., it draws with an hpen). Moreover, the *Qsquigl* wave tapers only at its ending point, whereas *Zsquigl* tapers at both the starting and ending points. All three are variations of *Zdraw* used in Don Knuth's *cmbase*, and they rely on a subroutine *Zcomp* (which is pilfered wholesale from *cmbase*) to handle the trigonometric nitty-gritty involved.



Msquigl



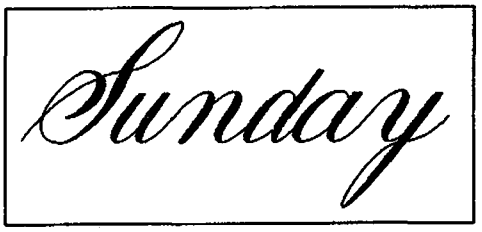
Tsquigl

Strokes which have a trough followed by a crest are produced by *Msquigl* or *Tsquigl*. *Msquigl* uses slightly thinner pens. Both subroutines require the following eight parameters: the wave's starting point (*start*), the x-coordinate of the turning point at the bottom of the trough (*trough*), the wave's midpoint (*midpt*), the x-coordinate of the turning point at the top of the crest (*crest*), the wave's ending point (*finish*), the index of the pen with which we start drawing, the index of the pen with which we conclude, and the slope at mid-wave. These subroutines are also adapted from *cmbase*. They call *Scomp* for computing values.

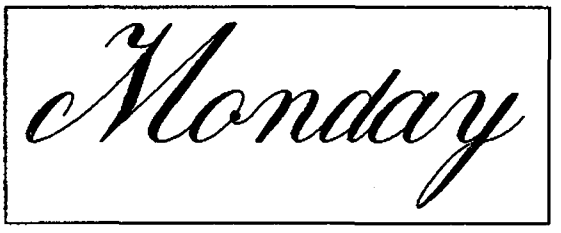
At one point, Kaufman and Homelsky remark, "The joining of letters into words is as important as the forming of the letters themselves". This nicely sums up another problem in designing *CuPt*, i.e., contriving to make discrete characters appear to join like flowing script. This is relatively simple for the lower case letters. I simply designed each so that the "beginning" of the character (that is, its left side) included the point with coordinates $(0,2/5h)$, and the "ending" of the character (that is, its right side) was precisely at the point with coordinates $(r, 2/5h)$. Occasionally, this produces a small bit of overlap:

The finishing upswing on the 'v' runs into the left side of the 'e', the finishing upswing on the 'r' runs into the left side of the 'l', and so on. Normally, this isn't apparent (I hope):

No single rule for beginning and ending points applies to the upper case letters, but then, none needs to: the upper case letters never need to be connected in sequence. What the precise value of an upper case letter's smallest x-coordinate is doesn't matter (as long, of course, as it is greater than zero). Some upper case letters (viz., H, J, K, M, R, U, X and Z) end in a hairline upswing that must connect to minuscules; for these, the right side must end at the point with coordinates $(r, 2/5h)$. The other upper case letters stand independently, and their right hand sides need only end at some point with an x-coordinate less than r.

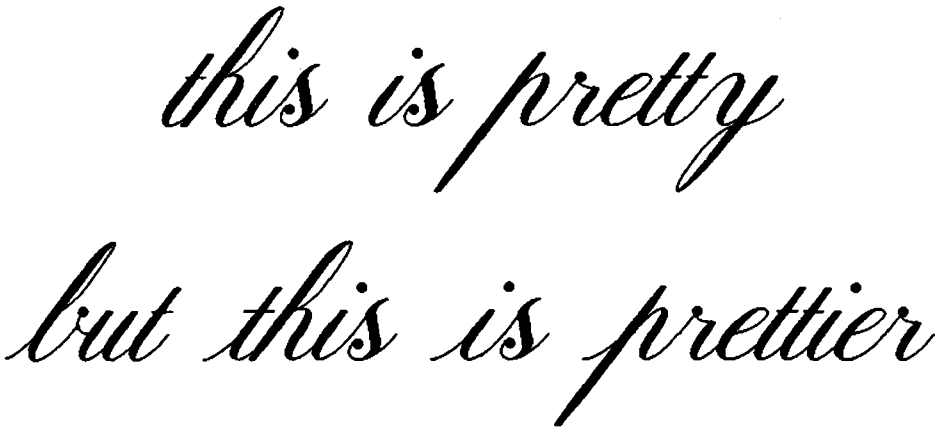


Non-Connecting Upper Case



Connecting Upper Case

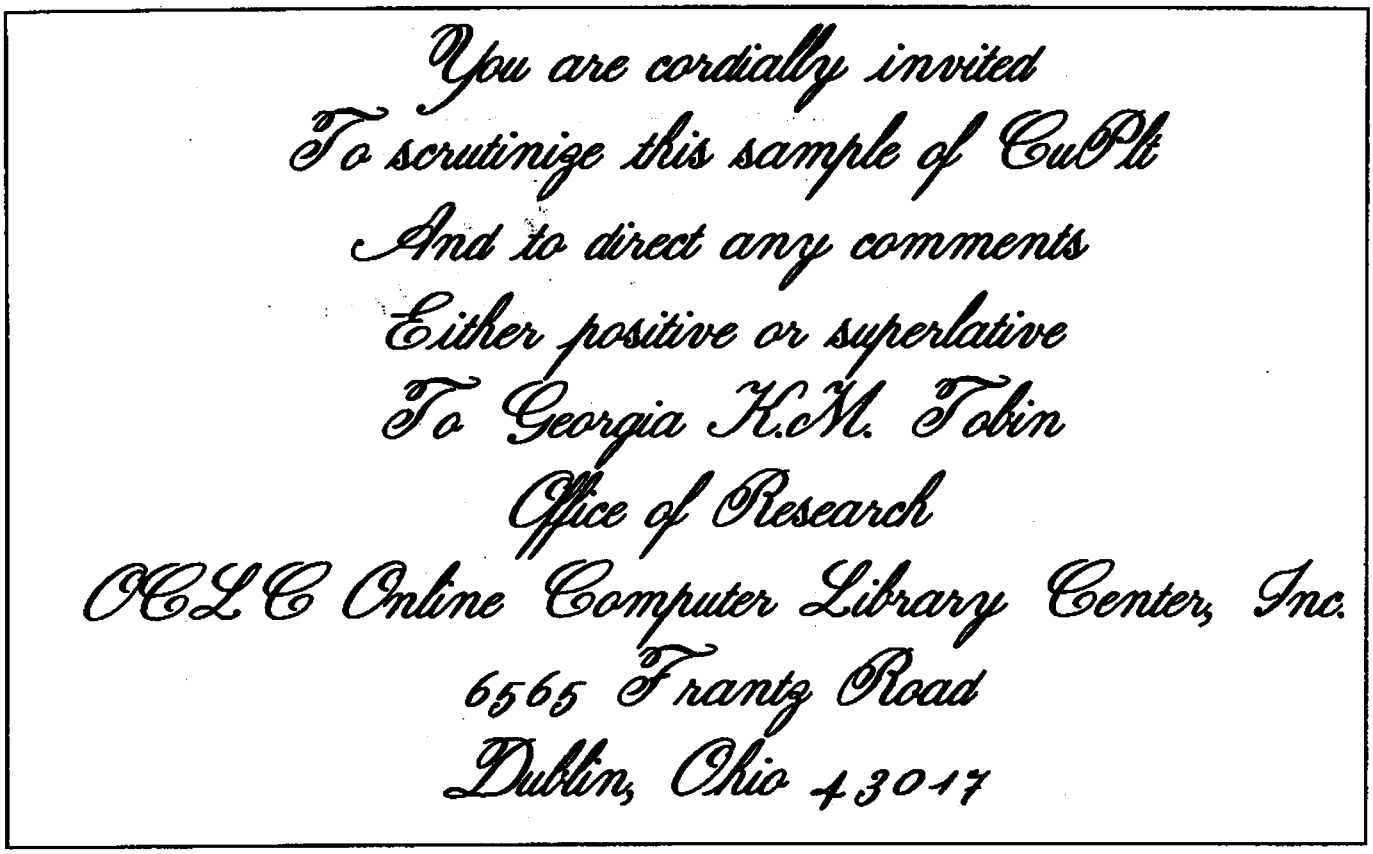
There are still a number of rough spots in *CuPt*. One problem is tied to the convention that the beginning of lower case letters include the point with coordinates $(0,2/5h)$. In certain cases, viz., b, h, i, j, k, l, p, r, s and t, the letter form which connects nicely is not always the most desirable form. When these letters do not follow another lower case letter, a form which begins with a lead-in hairline stroke from the baseline looks better than the usual combining form. That is, I claim that:



As of this writing, I have not found a way to put the information that (for instance) '001 is to replace 'b after a blank, or after certain upper case letters, or after certain punctuation marks. I am able to print these ligatures by manually inserting a dummy character (*) in front of letters to be replaced and including '* in the ligature table; but this approach is far too cumbersome and I hardly ever insert an * every place I need one.

At present, all upper and lower case letters, all digits and about a dozen punctuation marks are designed, and look acceptably good in *most* design sizes for *most* boldnesses and *most* expansions. I hope to carry out exhaustive testing to determine specific problems and to correct those, so that the same can be said of *all* design sizes at *all* boldnesses and expansions.

Meanwhile, here's a sample:



Bibliography

Hickey, Thomas B. "The Status of METAFONT at OCLC". *TUGboat*, Volume 2, No. 2, July 1981, pp. 35-38.

Hickey, Thomas B.; Tobin, Georgia K.M., *The Book of Chels*. Privately produced limited edition. 1982.

Kaufman, Herb; Homelsky, Geri. *Calligraphy in the Copperplate Style*. Dover Publications, Inc. New York, 1980.

Knuth, Donald E. *The Computer Modern Family of Typefaces*. Computer Science Department Report No. STAN-CS-80-780. January, 1980.

Knuth, Donald E. TEX *and* METAFONT*: New Directions in Typesetting*. American Mathematical Society/Digital Press. 1979.