

Post-Processing SAS/GRAPH® generated CGM's to Eliminate Text String Bugs

Gerhardt Pohl, Eli Lilly and Company, Indianapolis, IN

ABSTRACT

CGM (Computer Graphics Metafile) is a frequently used non-proprietary standard for transferring graphics between software packages. SAS® technical support document TS-252x recommends CGM as a format for transferring graphics between SAS/GRAPH software and Microsoft Word 97. Unfortunately, SAS/GRAPH generated CGM's suffer from a number of errors with regard to text strings: 1) Angled strings for y-axis labels are distorted 2) Strings are often clipped by graphics boundaries or bump into other objects 3) Characters in strings produced by the clear text device CGMCLEAR have incorrect aspect ratios. A technique for correcting these bugs by post-processing SAS/GRAPH generated clear text files using the DATA step as an automated text editor is discussed. In addition, the technique is used to insert other features into the CGM. For example, the CGM is given real size so that the graphic can be imported into Microsoft Word with the correct size specified by the HSIZE and VSIZE GOPTIONS.

INTRODUCTION

SAS/GRAPH graphics often must be embedded in documents of other applications. CGM provides a transport format which is widely implemented by many of these packages. It is vector based format rather than bit mapped so that it maintains the integrity of graphic objects such as lines. It has provisions to help control the mapping of fonts. In general, it is a relatively compact transport format with adequate richness for technical illustrations. By far its most appealing feature is its wide use. CGM import filters exist for most word processing and presentation packages.

SAS/GRAPH provides a great deal of support for CGM. Many different devices have been configured to address specific needs of different importing packages. Unfortunately, users often experience difficulties with SAS/GRAPH generated CGM's particularly with regard to text strings. This paper focuses on details of the CGM standard and how it is implemented by SAS/GRAPH version 6.12. The primary example of an importing package is taken to be Microsoft Word 97 with its import filter CGMIMP32.FLT developed by Henderson Software Inc.

BACKGROUND

CGM Formats

CGM can be written in three formats, binary, character and clear text. All three are supported by SAS/GRAPH. Binary and character formats provide the smallest files but these files are not easily readable by human beings. Clear text, as the name implies, has an English-like syntax directly readable by humans. Much of the discussion in this paper will focus on clear text CGM as it allows one to directly see how SAS/GRAPH implements CGM.

SAS/GRAPH Hardware and SoftwareText

SAS/GRAPH supports two types of fonts, software and hardware. Software text is simulated or drawn text, i.e., the text is written to the CGM as a series of graphic drawing primitives, lines, polygons, etc. Use of software fonts allows complete flexibility in what can be rendered. One can even define custom font sets via the GFONT procedure. The disadvantages of software fonts are that text is noticeably grainy and jagged when enlarged and it cannot be edited as text in the receiving package. The receiving package sees the small pieces of individual characters as essentially unrelated objects. Furthermore, use of software fonts greatly increases file size.

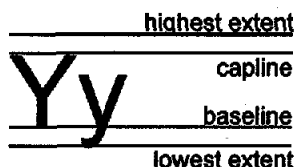
Hardware fonts have decided advantages over software fonts. Hardware text is written to the CGM as a true character string. This results in extremely small file sizes; and, most importantly, this allows the receiving package to retain the ability to resize and edit the text as a string.

PROBLEMS WITH SAS/GRAPH'S IMPLEMENTATION OF CGM

Incorrectly Sized Hardware Strings

SAS/GRAPH does not comply with the CGM standard for describing text size. In CGM, text height is properly measured from the baseline to the capline of capital letters. (See Figure 1.)

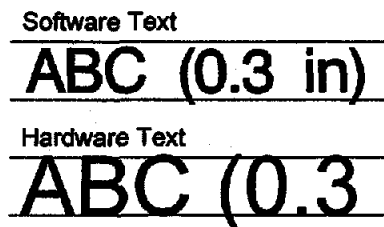
Figure 1:



SAS/GRAPH appears to measure text from lowest extent (the bottom of a "g" or "y") to the highest extent (typically the top of an apostrophe). This is the more common definition for text height as seen in the PostScript or Windows Metafile (WMF) formats. The result is that SAS/GRAPH draws software text smaller than expected under the CGM standard. The situation is worse for hardware text. It appears that when laying out graphics, SAS/GRAPH reserves space for hardware text as though it will be realized at the smaller size. When an importing package interprets a SAS/GRAPH CGM, the text is larger than anticipated and often overwrites other objects or is clipped by the graphic boundaries.

This behavior is demonstrated in Figure 2. For this figure, identical software and hardware strings were generated using the same SAS/GRAPH specifications for text height (HSIZE=0.3 in). Note that the software text (FONT=SWISS) fails to fill the space between the capline and baseline drawn 0.3 inches apart. The hardware text (FONT=HWCGM001), on the other hand, is realized by the importing software (MS Word 97) at the full size specified by the CGM standard. This results in the string being clipped by the right graphic boundary. Note that in preparing this example, SAS/GRAPH failed to detect that the hardware string extended beyond the boundary and no warning was issued to the log.

Figure2:



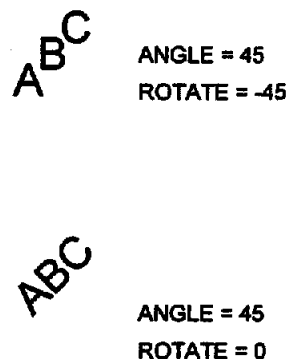
Readers wishing to recreate Figure 2 are advised that it is necessary to use a device which supports both hardware and software text. This figure was generated using DEVICE=CGMOF97P.

Discontinuous Angled Hardware Strings

One of the most immediately noticeable errors in SAS/GRAPH generated CGM is distorted angled text on y-axis labels. The origin of this distortion is complex and requires some appreciation of the limitations of the CGM standard.

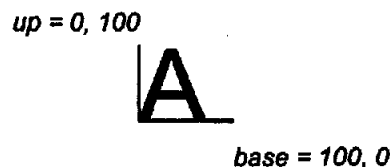
SAS/GRAPH supports a great deal of flexibility as regards character orientation. With the GOPTIONS of ANGLE and ROTATE, one is able to change both the orientation of the baseline of a text string and the rotation of each character relative to that baseline. (See Figure 3.)

Figure 3:



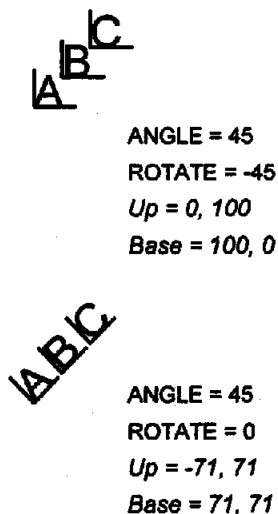
CGM supports character orientation largely through a single command *CharOri*. The parameters of the *CharOri* command are the percent coordinates of the "up" vector and the "baseline" vector for the string. (See Figure 4.) The magnitudes of these vectors control also the aspect ratio of the text; but we shall consider only their effect on direction, as SAS/GRAPH does not appear to use the *CharOri* command to effect the shape of text. Note importantly that CGM does not support arbitrary rotation of characters relative to the baseline.

Figure 4:



It appears that in order to support fully the richness of character orientations possible in SAS/GRAPH, the developers choose to use the direction of the *CharOri* to specify the ROTATE orientation. The ANGLE orientation is accomplished by simulating an angled baseline via a series of equally spaced individual character strings. (See Figure 5.)

Figure 5:



An excerpt of the clear text CGM output corresponding to Figure 5 is shown below to further emphasize that the angled text is accomplished via a series of separately placed strings.

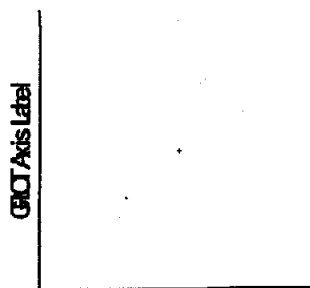
```
charori 0,100,100,0;
text 1214,8566 final "A";
text 1594,8988 final "B";
text 1974,9411 final "C";
...
charori -71,71,71,71;
text 1279,5000 final "A";
text 1524,5272 final "B";
text 1769,5545 final "C";
```

A better choice for use of the CGM vocabulary would have been to align the direction of the *CharOri* command to the orientation of the ANGLE. In this way, angled strings would have been output as single strings which would have been editable as single strings in the receiving package. Furthermore, the inter-character spacing of the string would have been accomplished using the full power of modern fonts. The coarse solution of equally spacing proportional hardware fonts often results in unreadable text with

inappropriate gaps or overwriting of adjacent characters. The only conditions under which the simulated spacing gives acceptable results are for non-proportional fonts (e.g., Courier) or occasionally for proportional fonts when only capital letters are used.

The unfortunate result of the developer's choice is that the most natural orientations, those with ROTATE=0 are accomplished using the most unnatural coarsely simulated solution. Consider, for example, the commonly occurring case of an angled y-axis label. Here the desired orientation is often ANGLE=90 and ROTATE=0. SAS/GRAPH severely distorts text of this type by breaking it into separate strings. (See Figure 6.)

Figure 6:



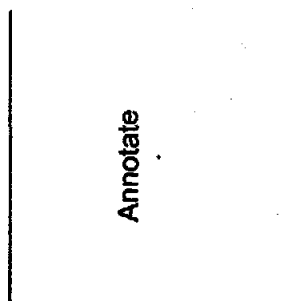
For emphasis, an excerpt of the clear text CGM output for Figure 6 is shown below. Note the separate strings simulating what should be a single angled a string.

```
charori -100,0,0,100
text 1697,4086 final "G";
text 1697,4352 final "P";
text 1697,4619 final "L";
text 1697,4885 final "O";
text 1697,5152 final "T";
text 1697,5418 final " ";
text 1697,5685 final "A";
text 1697,5951 final "x";
text 1697,6218 final "i";
text 1697,6485 final "s";
text 1697,6751 final " ";
text 1697,7018 final "L";
text 1697,7284 final "a";
text 1697,7551 final "b";
text 1697,7817 final "e";
text 1697,8083 final "l";
```

A curious exception to how SAS/GRAPH treats angled text is in the special case ANGLE=0. In this case only, the string is left intact and the *CharOri* baseline is set to match the ROTATE orientation as desired. Therefore, if one

deliberately miscodes text which one desires to have ANGLE=90 and ROTATE=0 as ANGLE=0 and ROTATE=90, one achieves a correctly angled intact string. (See Figure 7.)

Figure 7:



```
charori -100,0,0,100;
text 5361,5693 final "Annotate";
```

Unfortunately, this fix is effective only when using the ANNOTATE facility. When using the AXIS statement directly, the GPGLOT procedure will reserve too much room for the string forcing the y-axis far to the right. (See Figure 8.)

Figure 8:

GPGLOT Axis Label

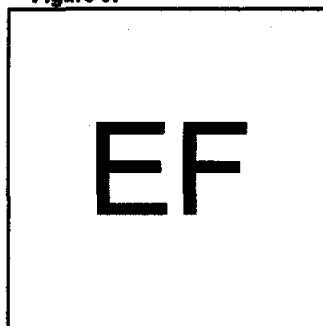
To reiterate, it seems that a better option for the developers of the SAS/GRAPH CGM devices would have been to execute only a subset of SAS/GRAPH character orientations well (those with ROTATE=0) rather than to simulate all possible orientations poorly.

Distorted Aspect Ratio of Individual Characters

A subtle error discovered only in output from DEVICE=CGMCLEAR is a distortion in the aspect ratio of individual characters. In CGM the aspect ratio of individual characters can be controlled using the *CharExpan* command. (See Figures 9 and 10.) A *CharExpan* value of 1 does not distort the shape of the characters. A value less

than 1 horizontally shrinks the characters. A value greater than 1 horizontally stretches the characters.

Figure 9:



```
charexpan 1;
```

Figure 10:



```
charexpan 0.5;
```

It is difficult to imagine scenarios in which a *CharExpan* value other than 1 would be useful; yet, in CGMCLEAR, SAS/GRAPH links the value of *CharExpan* to the aspect ratio of the entire figure, i.e., it appears that *CharExpan* is set equal to VSIZE/HSIZE. For perfectly square graphs this causes no distortion; however, for all other graphics the shapes of the characters are distorted. Consider, for example, Figures 9 and 10. These graphics were created with precisely the same specifications for character height using the GSLIDE procedure. The only difference in the graphics is the overall aspect ratio as set using the VSIZE and HSIZE options. The aspect ratio is 1:1 in Figure 9 and 1:2 in Figure 10. Note the difference in the length of the strings in the two figures and the thinning of the vertical lines in Figure 10.

Lack of Metric Scaling Option

An often encountered difficulty when importing SAS/GRAPH generated CGM's into other packages is that the sizes specified by the VSIZE and HSIZE GOPTIONS are not respected. The source of this difficulty lies in the fact that SAS/GRAPH makes no provision for giving CGM's real size.

Coordinates in CGM are based on an arbitrary grid system of virtual display coordinates (VDC). This grid system can be given real physical meaning via the CGM *ScaleMode*

command. The legal values for the *ScaleMode* are "abstract" and "metric." Abstract CGM's have no absolute size associated with them as it is anticipated that the graphic will be stretched to fit various viewing environments. This is the default mode in CGM when no *ScaleMode* is specified. Metric CGM's have real physical meaning. If the *ScaleMode* is metric, a second parameter is included in the CGM which specifies the number of millimeters per virtual display coordinate. Whether or not VSIZE and HSIZE are specified by the user, SAS/GRAPH software either omits the *ScaleMode* command completely or declares the *ScaleMode* to be abstract.

Importing packages faced with an abstract CGM respond in a variety of ways. The import filter in Microsoft Word Version 7 expands abstract CGM's until they reach the current setting for the page margin. In Microsoft Word 97 the filter responds by restricting abstract CGM's to a 4 inch by 4 inch box. The graphics must then be resized by the user to the original creation size via formatting the translated WMF within Word. In contrast, the filter in Microsoft Word 97 respects the size of CGM's with metric *ScaleMode*, i.e., metric CGM's will import immediately at the correct size. Furthermore, the filter sets the WMF original size parameters to the size specified in the metric CGM.

This is a very useful feature which SAS/GRAPH developers have not exploited.

Miscellaneous Minor Errors

A minor annoyance in SAS/GRAPH clear text CGM's is the improper specification of the values for the parameters of two CGM commands *MarkerSizeMode* and *LineWidthMode*. Absolute mode for these commands is properly written as "abs" rather than "absolute." This misspelling causes warnings in some import filters such as CGMIMP32.

THE POST-PROCESSING SOLUTION

Why Post-Process Rather than Change the Original Code

Errors such as clipped text resulting from SAS/GRAPH incorrectly sizing text strings cannot be corrected reliably by changes at the coding stage. To see this, consider the order in which the clipping occurs. 1) The programmer specifies a text size, say one unit. 2) SAS/GRAPH reserves space for the text under the misconception that it will be realized at 0.75 units. 3) The receiving program interprets the text at the full size 1.0 units resulting in clipped text. Intervention at step one, by the programmer specifying smaller text is followed by the errant step two in which SAS/GRAPH reserves still less space for the text perpetuating the error. Clearly, the

only ways to circumvent such problems are to correct SAS/Graph software itself so that it will reserve the proper amount of space or to intervene after SAS/GRAPH has made its errors in step two. In this example, reducing the font size specification in the CGM to match the size assumed by SAS/GRAPH.

Using the DATA step to Post-Process

Post-processing the CGM's can be accomplished using a variety of packages. Using the DATA step to perform the corrections assures that the necessary package to perform the work will always be available to the creator of the graphic. In addition, correcting the CGM directly from within SAS immediately following creation simplifies the transfer to destination packages. The impression left with the user is that the initial SAS/GRAPH CGM output was produced error free. No other intermediate packages need be invoked.

A SAS macro CGMFX2 was developed to correct the errors discussed in this paper. The macro is available from the author via e-mail.

To use this fix, one simply calls the macro after creating a CGM using the clear text device CGMCLEAR. Parameters to CGMFX2 are the name of the original CGM output file and the name of the output file in which to place the corrected CGM. These files can be the same in which case the original CGM is overwritten.

Clear text CGM was taken as the target on which to perform the corrections both to ease development of the macro and to provide the highest level of documentation while using the macro. Erroneous lines in the original CGM are retained as non-executable comments in the corrected CGM and new lines are marked. In most environments, the added file size of clear text CGM is not an issue as the graphic is often translated to other formats on import into the destination package. For example, upon insertion into Microsoft Word 97, the CGM is translated to WMF. Consequently, the size of the final document is the same whether the CGM transport format was binary or clear text.

Using the DATA step to process the clear text CGM's is not without technical challenges. These stem from the limitations SAS software places on the size of character variables (200 characters) and the basic orientation of the DATA step towards processing single records. As currently implemented CGMFX2 uses a single character variable to hold each line of the original SAS/GRAPH CGM when parsing it into single commands. This requires the original CGM to have line lengths no greater than 200 characters. Users are advised to create the CGM using GOPTIONS GSLEN=200.

Repairing Incorrectly Sized Strings

Correcting clipping due to incorrectly specified text height is easily accomplished. CGMFX2 searches the original CGM for all instances of the *CharHeight* command and scales the original height down by 25% to approximate the size of software text. Excerpts from CGMFX2 output are shown below:

```
% charheight 595; %  
% NEW LINES * %  
charheight 446;  
% END NEW LINES %
```

Note that the percent character is the comment delimiter in clear text CGM. The command CHARHEIGHT 595 is a line from the original CGM which has been commented out by CGMFX2 so as to be non-executable. Between the NEW LINES and END NEW LINES comments is a new line which reduces the size of the text.

Repairing Discontinuous Angled Strings

Repairing improperly rendered angled strings is the most difficult aspect of correcting SAS/GRAPH CGM's. As currently written the macro only attempts to repair strings that were written with ANGLE=90 and ROTATE=0. It was thought to be more useful to have CGMFX2 automatically identify all such improperly rendered angled strings than to put the burden on users to correctly pass the content of all such strings to the macro as parameters. CGMFX2 uses an algorithm based on several factors to identify the improperly rendered strings. First the target text for correction must be have a character orientation of ninety degrees to the horizontal. Second the individual text strings to be concatenated must be no longer than one character; and, finally, the text strings must have approximately the same x-coordinate and a y-coordinate no more than one character height away. This is a rather generous window in the y direction equivalent to about two spaces. The reason the x-coordinate has been taken to be only approximately the same (± 10 VDC) is that SAS/GRAPH generated CGM's have been observed in which the characters in the improperly rendered string inexplicably have x-coordinates that differ slightly from one another.

An excerpt of CGMFX2 output showing a repaired y-axis label from the Gplot procedure is shown below:

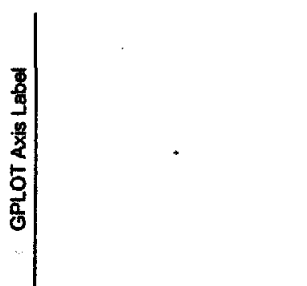
```
% text 1697,4086 final "G"; %  
% text 1697,4352 final "P"; %  
% text 1697,4619 final "L"; %  
% text 1697,4885 final "O"; %  
% text 1697,5152 final "T"; %
```

```
% text 1697,5418 final " "; %  
% text 1697,5685 final "A"; %  
% text 1697,5951 final "x"; %  
% text 1697,6218 final "i"; %  
% text 1697,6485 final "s"; %  
% text 1697,6751 final " "; %  
% text 1697,7018 final "L"; %  
% text 1697,7284 final "a"; %  
% text 1697,7551 final "b"; %  
% text 1697,7817 final "e"; %  
% text 1697,8083 final "l"; %  
% NEW LINES %  
text 1697,6084 final "Gplot Axis  
Label";  
% END NEW LINES %
```

Note that in addition to concatenating the individual characters into one string, CGMFX2 recalculates the position of the string taking the y-coordinate to be the average of the y-coordinates of the individual characters.

The graphic resulting from applying CGMFX2 to the CGM which produced Figure 6 is shown in Figure 11. Note that the size of the text in Figure 11 is smaller than that in Figure 6 as discussed in the previous section.

Figure 11:



Repairing Distorted Aspect Ratios in Characters

Distorted aspect ratios are fixed by replacing all *CharExpan* commands with aspect ratios other than 1 with a *CharExpan* of 1. Excerpts from CGMFX2 output are shown below:

```
% charexpan 1.6; %  
% NEW LINES %  
charexpan 1;  
% END NEW LINES %
```

Adding Metric Scaling

CGMFIx2 adds metric scaling to the CGM with the value for the size parameter set to correspond to `DEVICE=CGMCLEAR`. One may verify using the `GDEVICE` procedure that `CGMCLEAR` has `XPIXELS=25320` by `YPIXELS=32767` with a total graphics output area of 8.5 inches by 11.0 inches. Converting this to millimeters per pixel yields approximately 0.0085264. This is the value used in the added line. Users wishing to adapt this technique to other devices should note that this value differs from device to device depending on the device properties. CGM's with this line inserted show true correspondence between all sizes (`HTEXT`, `HSIZE`, `VSIZE`, etc.) specified in `SAS/GRAPH` and those seen after import into Microsoft Word 97. An excerpt from `CGMFIx2` output is shown below:

```
% NEW LINES %
scalemode metric .0085264;
% END NEW LINES %
```

CONCLUSION

Post-processing `SAS/GRAPH` output files is a successfully implemented method for correcting problems encountered in `SAS/GRAPH` CGM's. A better solution, of course, would be to have the errors removed from `SAS/GRAPH` software. The author is happy to report that private communication with the support staff at SAS Institute® suggests that some of the problems discussed in this paper will be addressed in the upcoming version 8.2. Until that time, it is hoped that a better understanding of how CGM is implemented in `SAS/GRAPH` will allow users to achieve better results in their graphics.

REFERENCES

"EXPORTING `SAS/GRAPH`® Output to Microsoft Word, Excel and Powerpoint 97," TS252X-2nd Edition, SAS Institute Inc. (11/1/99).

ACKNOWLEDGEMENTS

The author gratefully acknowledges the contributions of Alex Dmitrienko at Eli Lilly and Company for insights and examples of errant `SAS/GRAPH` CGM's.

SAS, SAS Institute, and `SAS/GRAPH` software are registered trademarks or trademarks of SAS Institute, Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Gerhardt Pohl, Ph.D.
Eli Lilly and Company
Lilly Corporate Center
Drop Code 2233
Indianapolis, IN 46285
317-277-5113
pohl_gerhardt@lilly.com