

SOME THOUGHTS ON MAKING DSGI USER-FRIENDLY

by Jim Comer, Sr. Programmer Analyst
Glaxo Inc., Research Triangle Park, NC

Around a year ago, we were looking for a way to improve the appearance of Case Report Form Summaries for the Respiratory group of the Biostatistics Department at Glaxo. The results of serial pulmonary function tests and diary symptom scores are easier to understand when presented as a graph rather than as a table, so we needed to find a way to combine output from SAS/Graph procedures with text generated from our SAS datasets. After evaluating several alternatives, the Data Step Graphics Interface seemed to be the most straightforward approach. DSGI proved to be a very powerful tool that can substantially enhance the quality of output generated by SAS, but it can also be very difficult to learn and intimidating to use.

DSGI BASICS

DSGI is modeled after the Graphics Kernel Standard. It consists of five functions and one call routine.

GINIT is the function that opens DSGI.

GSET is the function used to set various attributes that are used when drawing on the output page. There are around 50 different attributes that can be set, such as aspect ratio, text alignment, fonts, colors, catalog, etc.

GASK is a call routine that is used to check the current settings of the various attributes.

GRAPH is the function that is used to manage the graph catalogs used with DSGI.

GDRAW is the function that is used to draw graphics elements on the output page or print messages on the SAS log file. There are 10 different graphic elements that can be drawn, such as lines, arcs, text, etc.

GTERM is the function used to close DSGI.

There are four operating states that are set in DSGI by calls to the various functions. Additionally, each of the functions can be called only during the proper operating state. For example, you must initialize

DSGI before attempting to use the GDRAW function.

Typically, a program using DSGI will have the following structure:

Begin by opening DSGI using the GINIT function.

Next, the windows used in the program must be defined. At least one window is required but up to 21 may be defined. The GSET('VIEWPORT',...) function is used to define the position of the window on the physical page. The coordinate system that will be used in that window is established with a call to the GSET('WINDOW',...) function. Windows may overlap, so it is possible to write text in a window using one coordinate system and then draw graph elements using a different coordinate system and have the results written to the same physical position on the output page.

Once the windows have been defined, initialize the current graph page with the GRAPH('CLEAR',...) function.

Activate one of the windows using the GSET('TRANSNO',...) function and write any graph elements, including text, into the selected window using the GDRAW(...) function. Existing graph segments created by SAS/GRAPH procedures such as GPLOT or GCHART may be imported into the window using the GRAPH(...) function.

Continue activating the various windows until everything has been written to the output page. Call the GRAPH('UPDATE',...) function to output the newly created graph segment and write it to the current graphics stream file.

Once the output page has been written, either re-initialize the workspace using the GRAPH('CLEAR',...) function or close DSGI using the GTERM function.

DIFFICULTIES AND INTIMIDATIONS

Although we were impressed with the power and flexibility of DSGI as compared to the alternatives,

there are a number of things about DSGI that appear to be somewhat alien to the typical SAS user.

The first thing about DSGI is that the X coordinates increase from the bottom of the page toward the top. This is normal when plotting a graph, but when generating reports, most users are more comfortable counting lines from top to bottom.

Instead of the relatively simple PUT statement, DSGI uses function calls with a varying (and sometimes excessive) number of positional parameters to generate the output page. For example, the call to draw a box requires 12 parameters and the call to return the length of a text string requires 15 parameters.

Another issue is the use of default values. Many users prefer to go with default values unless they absolutely have to set something. With most DSGI routines, not only are there no defaults, but if the variables used with them are not properly initialized, the job will fail.

DSGI can be difficult to learn because it is relatively complex. The routines must be called when the DSGI system is in the proper state, which is set by previous function calls. Some of the features require several different calls to set various attributes before the feature may be used to produce output.

Finally, the error messages generated by DSGI can be rather cryptic, making it difficult to debug a program that is failing. When DSGI is active, the PUT statement is deactivated, so the old standby of writing messages to the SAS log to help debug a program becomes a little more complicated. The GDRAW('MESSAGE',...) function must be used to write text to the log file.

MACROTIZE IT!!!

To make the power of DSGI available to the typical user in our department, we decided to develop a series of macros that automate much of the overhead required by DSGI. The macros create a coordinate system that numbers lines from top to bottom; provide 'global' variables that keep track of the current line and column position as well as settings for font, line height, line spacing, etc.; establish default values for almost all of the features

of DSGI used in the macro library; and perform a reasonable level of error checking. The macros do not attempt to address all of the features available with DSGI, but all of the features that are commonly used in our department have been implemented.

There are six major categories of functionality in the macro library we have implemented:

DSGI SYSTEM CONTROLS are used to open and close the DSGI system, set the appropriate GOPTIONS for the job, establish the number of rows and columns on the page as well as the page orientation, create the 'global' variables that are used to keep track of the current settings, and establish many of the default values used by the macros.

%DGINIT	opens the DSGI system.
%PSETUP	sets margins and other attributes for portrait output.
%LSETUP	sets margins and other attributes for landscape output.
%DGCLOSE	closes the DSGI system.

GRAPH IMPORT macros are used to initialize windows and retrieve entries from a SAS/Graph catalog and insert them in the specified window.

%GDEF	defines a window and viewport for a graph.
%PUTGRAPH	inserts an existing graph segment into the window.
%BOXGRAPH	draws a frame around a graph window.

FONT SELECTION macros are used to select the active font (either hardware or software), as well as the height and justification of text generated using the active font.

%SETFONT	selects active font.
%SETHGT	sets the height of the text relative to the current coordinate system.
%SETJ	sets text justification left, center or right.

POINTER POSITIONING macros are used to move to a new location on the page, establish margins and output a page when it is complete.

%SKIP	skips to a new line.
%NEWPOS	moves to a specified row and/or column.

%RELMOVE moves to a new row and column relative to current position.
%SETMAR sets margins.
%NEWPAGE outputs page and prepares a new page.

TEXT PROCESSING AND SPECIAL EFFECTS

macros are used to write text, lines and shaded boxes on the page. Additional features include routines that determine the length of a character string on the output page; wrap character strings and flow them on the page.

%GLEN returns length of a text string.
%GWRAP wraps text strings according to length.
%GFLOW flows text on the page between the current column and the right margin. Remaining text is written between the left and right margins, beginning a new line as necessary.
%GPUT writes text to the output page.
%PUTSUB writes subscripted text to the output page.
%PUTSUP writes superscripted text to the output page.
%GMARK draws a specified marker on the page.
%DRAWLN draws a line between two points.
%BOX draws a box given row and column of opposite corners.
%BANNER writes up to 5 lines of text in a shaded box.

UTILITY routines are included to do error checking and general housekeeping functions. They are intended to be called by the other macros in the system and are not documented for the typical user.

BENEFITS

The DSGI macros have made it possible for the typical SAS user in our department (a statistician with little or no formal training in computer programming), to effectively use the power of DSGI. The macro calls look like the SAS statements that they are accustomed to using, so it's easier for them to get a program coded. For example, the macro call:

```
%GPUT('This is the output string',5,5);
```

looks a lot more like the SAS statement:

```
PUT #5 @5 'This is the output string';
```

than does the equivalent DSGI statement:

```
RC = GDRAW('TEXT',5,60,  
'This is the output string');
```

The level of error checking performed and the error messages that are generated by the macros make it much easier to develop new programs that use DSGI. The net result of using the macros is that the quality of the output has been substantially improved while the amount of time spent writing the programs has been minimized.

PERILS AND PITFALLS

While DSGI is a very powerful tool for generating quality output and it is one of the best ways to put text and graphs on the same page, it is also a resource hog. A tremendous amount of disk space is required as SAS generates a catalog entry for each page written to the Graphics Stream File (GSF). This is in addition to the catalog entries generated by SAS/Graph procedures that are included in the DSGI output.

We have minimized the disk storage requirements on our VAX system by using hardware fonts whenever possible, but one program that produced a safety listing for a recent NDA submission generated 250 megabytes of PostScript output.

It can also take a lot of CPU time and clock time to execute and print a complicated DSGI program. A program that generated a 3 to 5 page Case Report Form Summary for 350 patients used one hour of CPU time on a VAX 9000. The program was submitted at 8:00 am and it was still executing at 5:00 pm. It took over four hours to print the output file on a PS-40. The moral is plan ahead to get your program finished at least two days before your deadline.

One of the most difficult problems that we encountered in developing the system of macros is getting an accurate assessment of the length of a character string. When using the software fonts provided by SAS, the length can be determined exactly every time; however, the length returned when using the hardware fonts and the PS300 device driver is sometimes exact, sometimes a

reasonable approximation, and sometimes a shot in the dark. The SAS Institute has acknowledged this problem as a bug in the device driver. They are hoping to get it corrected soon.

APPLICATIONS

Although originally intended to be used primarily as a report generating tool, the DSGI macros have gotten varied use in the department. In addition to reports, they have been used to generate forms and presentations. Here are some examples:

Sample Output from the DSGI Macros

THE BANNER MACRO ACCEPTS UP 5 LINES OF MESSAGES
 OPTIONS ARE AVAILABLE TO CONTROL
 JUSTIFICATION OF THE TEXT
 SHADING OF THE BACKGROUND
 AND LINE WIDTH OF THE SURROUNDING BOX

THE MACROS ALLOW YOU TO EASILY DO _{SUB}SCRIPTS AND ^{SUPER}SCRIPTS

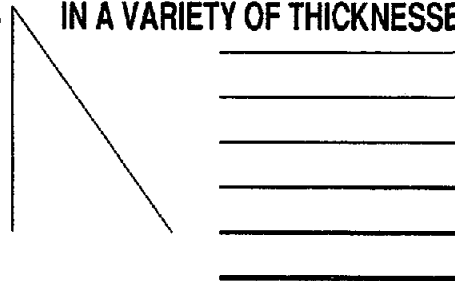
LONG LINES OF TEXT CAN EASILY BE WRAPPED TO FIT AVAILABLE SPACE

FOR EXAMPLE:

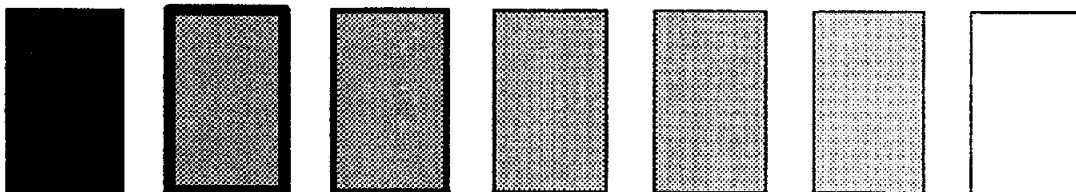
LONG LINES OF
 TEXT CAN EASILY
 BE WRAPPED TO
 FIT AVAILABLE
 SPACE

THE LINE DRAWING MACRO ALLOWS

VERTICAL, HORIZONTAL, OR DIAGONAL LINES IN A VARIETY OF THICKNESSES



THERE IS ALSO CONTROL OVER SHADING INSIDE BOXES



AND LINE WIDTH AROUND BOXES

For PostScript output to a file, the following hardware fonts a

Hardware font 01 is COURIER

Hardware font 02 is COURIER ITALIC

Hardware font 03 is COURIER BOLD

Hardware font 04 is COURIER BOLD ITALIC

Hardware font 05 is TIMES-ROMAN

Hardware font 06 is ***TIMES-ROMAN ITALIC***

Hardware font 07 is **TIMES ROMAN BOLD**

Hardware font 08 is *TIMES-ROMAN BOLD ITALIC*

Hardware font 09 is HELVETICA

Hardware font 10 is **HELVETICA ITALIC**

Hardware font 11 is HELVETICA BOLD

Hardware font 12 is HELVETICA BOLD ITALIC

It is also possible to request any of the SAS software fonts, s

BRUSH, Script, or ZAPF

but in general, these should be avoided when generating output

The following markers can be added to output using the GMARK macro:

+ X * □ ◇ △ # ∴ · ● ○ ◻ ♀ ♥ ♦ ♣ ♠ ✱ ⊙ ☿ ♀ ⊕
♂ ♢ ♧ ♨ ♂ Ψ Ε Ϸ ⚡ *

Markers can also be produced with the **MARKER** software font:

Protocol: XXX-999
Investigator: Z012
Patient: X9999

Background Characteristics of Patients

Study Design : Parallel

Date of Birth : 16JUL50

Age : 37

Sex : Female

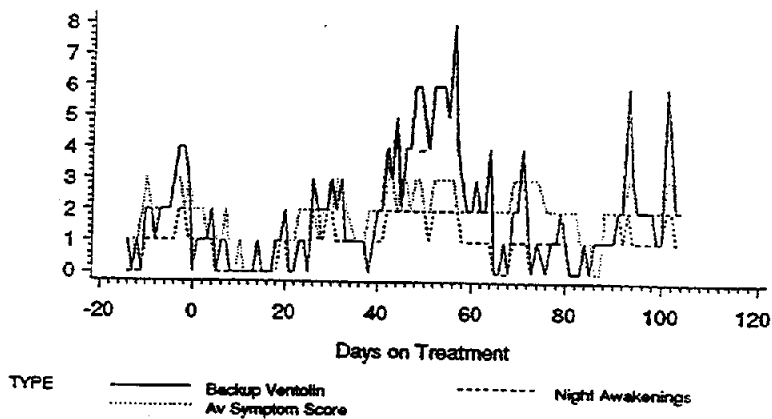
Height : 66 inches

Weight : 121 pounds

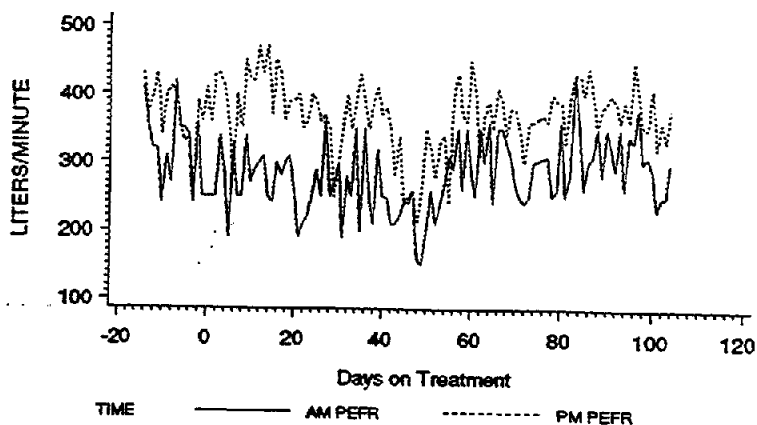
ECG

Period	Visit Date	Collection Interval	Result	QTc	Q Wave	Ventricular Rate	Minnesota Scores
1003	20MAY88		1	Normal	Normal	Normal	
1007	02SEP88		1	Normal	Normal	Normal	
1010	08JUN89		1	Normal	Normal	Normal	

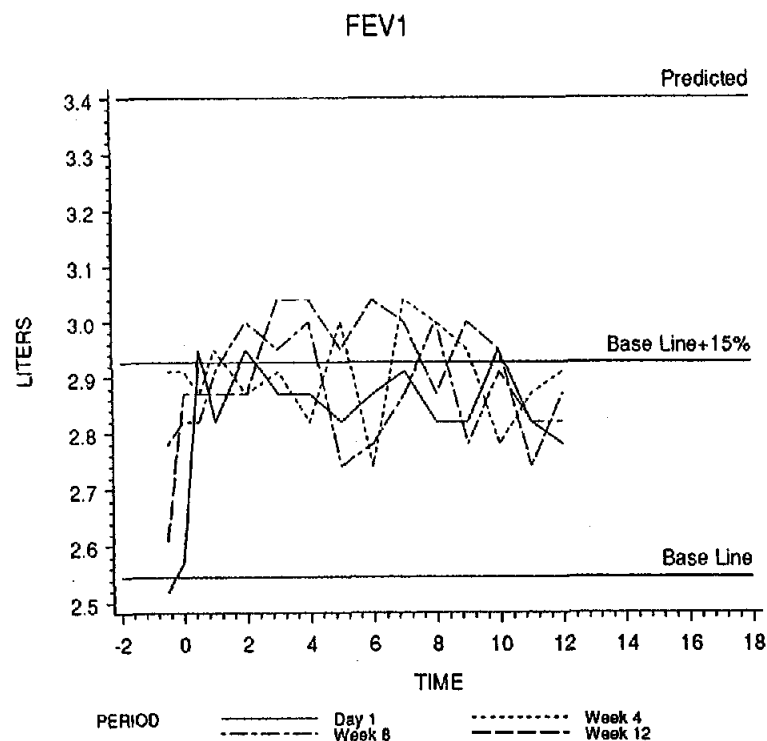
DIARY SYMPTOMS



PEFR



Protocol:
Investigator:
Patient:
Treatment:



Functions of Serial FEV1

	Day 1	Week 4	Week 8	Week 12
Time to Onset (hours)	0.47	0.85	+	+
Duration (hours)	1.82	0.44	+	+
Maximum Effect (% Improvement)	15.91	19.45	17.88	19.45
AUC over 15% (liter-hours)	0.00	0.01	+	+
AUC over baseline (liter-hours)	3.78	4.27	4.02	4.70

+ Indicates that patient did not achieve 15% improvement within one hour.

Diary Data: Weekly Averages

Week	AM PEFR	PM PEFR	Ventolin Use	Night Awakenings	Average Symptom Score
-1	422	425	0.0	0.0	0.0
0	414	417	0.6	0.0	0.2
1	419	417	0.0	0.0	0.0
2	419	419	0.0	0.0	0.0
3	414	427	0.0	0.0	0.0
4	417	416	0.0	0.0	0.0
5	406	416	0.0	0.0	0.3
6	417	431	0.0	0.3	0.0
7	419	420	0.0	0.0	0.0
8	421	433	0.0	0.0	0.0
9	417	424	0.0	0.0	0.0
10	411	409	0.0	0.0	0.2
11	390	396	0.3	0.0	0.3
12	404	409	0.0	0.0	0.0
13	383	394	0.0	0.0	0.3

Visit Assessments

Visit	Physician Assessment	Pulmonary Auscultation
Screening	No symptoms	No Wheezing
Day 1	No symptoms	No Wheezing
Week 2		No Wheezing
Week 4	No symptoms	No Wheezing
Week 6		No Wheezing
Week 8	No symptoms	No Wheezing
Week 10		No Wheezing
Week 12	No symptoms	No Wheezing

SAMPLE THEOPHYLLINE TITRATION LOG

Protocol 	Inv. No. 	Pt's. Initials 	Pt. No. E TXX
---	---	--	--

Theophylline: NO

Visit	Date month day year	Time (24 hr clock)	Check if Level is Undetectable	Level or Limit	# capsules BID
2			₁		2
2A			₁		2
2B			₁		2
2C			₁		2
4			₁		
6			₁		
8			₁		
10			₁		
Early Termination			₁		

KMK: 13JUN91