ABSTRACT:

This paper will show two methods for printing more than one graphic output on a page. Also, the paper will show the ability to surpass the realm of PROC GREPLAY and to end with the unexpected ability to create customized diagrams with the DATA Step Graphics Interface (DSGI), to meet the standards of critical viewers. Still, GREPLAY is suited for quick and easy use through its windowing features.

The objective of this paper is to explain and illustrate, the differences between these two tools. This way one knows the advantages and limitations to consider when selecting one versus the other as a tool for replaying graphic.

INTRODUCTION:

There exist two methods in SAS which allow the user to play back graphs stored in graphics catalogs. These methods are PROC GREPLAY and the DATA Step Graphics Interface (DSGI) tool. Both methods are very similar, but DSGI has a greater capability to customize output. This paper will begin to discuss how the two methods are run, show a break down of abilities, and display various customization features.

The environment at G.D. Searle consists of a VAX cluster, using Digital Postscript printers. The work-stations consist of Mac and IBM PC's, and VAX terminals, with an emulation package connecting the PC's to the cluster.

BASIC WORKINGS OF PROC GREPLAY:

Definition: This procedure replays graphics output stored in SAS catalog entries and manages the entries within the catalogs. PROC GREPLAY also creates templates and color maps that can be used while replaying the graphics output. PROC GREPLAY has the ability to use up to four different catalogs at once, the catalogs are listed as follows, Input (IGOUT), Output (GOUT), Template (TC), and Color Map (CC) catalogs.

When running PROC GREPLAY, the user must remember that the procedure will continue to run until stopped. It is very important to remember to code a STOP, QUIT, or END statement with this procedure. Also when a SAS session is disconnected the procedure will be stopped. To simply demonstrate how PROC GREPLAY replays a graph, the following code was developed,

```
PROC GREPLAY IGOUT=libref.incat NOFS;
REPLAY graphicname;
RUN; QUIT;
```

the output is shown as follows in figure 1.

```
Figure 1
```

PROC GREPLAY, unfortunately does not have the ability to add headers and footers on its own, they must be previously designed within PROC GSLIDE or other graphic procedures. Let's next take a look at how DSGI would handle the above task.

BASIC WORKINGS OF DSGI:

Definition: The DATA Step Graphics Interface enables the user to create graphics output within the DATA Step, or from within a SCL application. Graphic routines used by SAS/GRAPH can be called to generate an entire custom graph or to add features to an existing graph not unlike the Annotate facility of SAS/GRAPH. An example of DSGI code similar to the previous example for PROC GREPLAY can be found on the following page,
DSGI ALLOWS GLOBAL TITLES TO BE USED WHEN REPLAYING A GRAPH

PROC GREPLAY

PROC GREPLAY can be either used in full-screen windowing mode, or with line or batch mode statements. The examples found in this paper will be using line or batch processing. However, the windowing feature for this procedure is very handy.

PROC GREPLAY has the ability to replay graphs and manage catalogs containing graphs. The managing features available in PROC GREPLAY help to keep the Input catalogs (IGOUT) clean and organized, and allows graphics output in a catalog to be retrieved easily. There are five tasks available for managing a catalog, Copy, Group, Delete, Modify, and Move.

**COPY** → copies a graphics output from the input catalog (IGOUT) to the output catalog (GOUT), only able to use if an output catalog (GOUT) is specified.

**GROUP** → arranges graphics output into logical groupings.

**DELETE** → deletes unneeded graphics outputs.

**MODIFY** → changes graphics output names and descriptions.

**MOVE** → reorders graphics output.

These managing features are very exciting and helpful, such as when the user is pulling catalog information for replay purposes, GREPLAY has the ability to keep the IGOUT catalog entries clean. If a catalog is written to more than once there will exist old and new output entries, this can cause inappropriate entries to be printed if the user is not careful. The DELETE statement, will allow only those wanted entries to be kept after every run.

An example of why managing features are helpful could be shown when running under production mode and your graphic output must be created weekly. As a user, you are not going to want to have to clean out the catalogs yourself, separate from the running of the program. It is also possible to use an "X" command to delete the entire catalog prior to generating new catalog entries. The following code contains a few of the above mentioned managing tasks.

**FOOTNOTE**

**TABLE**

**TITLE**
Now let's create a similar output with DSGI.

**ABILITIES, DSGI:**

DSGI has the ability to replay previously generated graphics output, and allows text and other graphics to be added to the output. However, DSGI does not have the ability to manage graphics output catalogs like PROC GREPLAY, so the user has to be careful as to what may exist in their graphics catalogs.

In order to insert a graphics output in DSGI, there must exist a catalog where graphs exist. A name can be associated with a graph which helps in identifying where to place specific graphics. A square coordinate system will define the space in which the graph will be placed, and finally, the statement to draw the graph.

When writing DSGI code there are five main steps which must be included. These five steps are as follows:

1. initialize DSGI (GINIT());
2. open a graphics segment (GRAPH('clear', ...));
3. generate graphics elements (GDRAW(...));
4. close the graphics segment (GRAPH('update', ...));
5. end DSGI (GTERM());

Once an attribute is set here, it remains until changed or code is ended. Also, once a DSGI graphics output is stored, it is possible to re-display the graphics with PROC GREPLAY.

DSGI uses a few of SAS/GRAPH’s global options, such as, TITLE, FOOTNOTE, and GOPTIONS statements. With this feature, it allows DSGI to insert its own global options without creating an extra graphic output. A similar output design as seen in PROC GREPLAY is as follows. Notice DSGI does not have the ability to manage SAS catalogs.

**FOOTNOTE1**

F=SPECIAL H=2 'J J J'
F=CENTX H=1.5 'A LIST'
F=WEATHER H=2 'C C C'
F=CENTX H=1.5 'B LIST'
F=SPECIAL H=2 'D D D'
F=CENTX H=1.5 'C LIST'
F=SPECIAL H=2 'A A A'
F=CENTX H=1.5 'D LIST';

**TITLE1**

F=CENTX H=3 'DSGI ABILITIES';

**DATA NULL:**

```sas
/ *** SET CATALOG WHERE OUTPUT EXISTS ***/
RC=GSET('CATALOG','libref','incat');
RC=GINIT();
RC=GRAPH('CLEAR','SCALE');
```

---

**Figure 3**

---

**PROC GREPLAY ABILITIES**

---

Now let's create a similar output with DSGI.
The output put for the above code is as follows in figure 4.

DSGI ABILITIES

Similar graphs can be generated with both graphics tools. So far both methods tend to be quite comparable, lets now take a look into the customization features of these tools.

CUSTOMIZATION AND OPTIONAL FEATURES OF PROC GREPLAY:

With PROC GREPLAY, customization is very limited, in fact, the only type of customization which could be done would be to use ANNOTATE prior to running PROC GREPLAY, and at that the Annotate data can not be called by PROC GREPLAY. The procedure allows the user to choose and design their own color maps (CMAP) and design their own Templates (TC) and that is about the extent of customization for this procedure.

Since GREPLAY customization features are so limited, there will not be an example in this section. There are more features available for TDEF not presented in this paper which might be of more help for overlaying window examples.

CUSTOMIZATION AND OPTIONAL FEATURES OF DSGI:

SAS/GRAPH’s DSGI has an enormous ability for customizing graphics. This ability was shown in a paper presented at SUGI 14.

DSGI allows the user to generate a variety of objects. The drawing elements are associated with the GDRAW function. The associated attributes are listed below:

1. ARCS
2. BARS
3. ELLIPSES
4. ELLIP ARCS
5. LINES
6. MARKERS
7. PIE SLICES
8. POLYGONS
9. TEXT
10. FILLED ENCLOSED AREAS.

with these objects, the user has the ability to specify color, pattern, size, style, and position of the graphics elements. The following is an example of using some of the above mentioned objects.

DATA DSGIDRAW;
   RC=GSET('CATALOG','libref','incat');
   RC=GINIT();
   RC=GRAPH('CLEAR','STAR');
   RC=GSET('COLREP',1,'GREEN');
   RC=GSET('COLREP',2,'BLUE');
   RC=GSET('TEXCOLOR',1);
   RC=GSET('TEXCOLOR','XSWISS');
   RC=GSET('TEXHEIGHT',5);
   RC=GDRAW('TEXT',55,90,'Filled Star');
   RC=GSET('TEXHEIGHT',4.5);
   RC=GDRAW('TEXT',47,85,'Created with DSGI');
   RC=GSET('LINCOLOR',2);
   RC=GSET('FILCOLOR',1);
   RC=GSET('FILTYPE','SOLID');
   RC=GSET('LINWIDTH',4);
   RC=GDRAW('LINES',2,35,55,55,55);
   RC=GDRAW('LINES',2,55,65,55,75);
   RC=GDRAW('LINES',2,75,65,55,75);
   RC=GDRAW('LINES',2,75,95,55,55);
   RC=GDRAW('LINES',2,80,95,40,55);
   RC=GDRAW('LINES',2,85,80,15,40);
   RC=GDRAW('LINES',2,85,65,15,30);
RUN;

Figure 4

with these objects, the user has the ability to specify color, pattern, size, style, and position of the graphics elements. The following is an example of using some of the above mentioned objects.

DATA DSGIDRAW;
   RC=GSET('CATALOG','libref','incat');
   RC=GINIT();
   RC=GRAPH('CLEAR','STAR');
   RC=GSET('COLREP',1,'GREEN');
   RC=GSET('COLREP',2,'BLUE');
   RC=GSET('TEXCOLOR',1);
   RC=GSET('TEXCOLOR','XSWISS');
   RC=GSET('TEXHEIGHT',5);
   RC=GDRAW('TEXT',55,90,'Filled Star');
   RC=GSET('TEXHEIGHT',4.5);
   RC=GDRAW('TEXT',47,85,'Created with DSGI');
   RC=GSET('LINCOLOR',2);
   RC=GSET('FILCOLOR',1);
   RC=GSET('FILTYPE','SOLID');
   RC=GSET('LINWIDTH',4);
   RC=GDRAW('LINES',2,35,55,55,55);
   RC=GDRAW('LINES',2,55,65,55,75);
   RC=GDRAW('LINES',2,75,65,55,75);
   RC=GDRAW('LINES',2,75,95,55,55);
   RC=GDRAW('LINES',2,80,95,40,55);
   RC=GDRAW('LINES',2,85,80,15,40);
   RC=GDRAW('LINES',2,85,65,15,30);
RUN;

1478
The output generated from this code is a filled star as seen below in figure 5.

![Filled Star
Created with DSGI](image)

**Figure 5**

DSGI has the ability to bring in a graphics output and add some character and design to meet those tough business standards of today and to make the graph stand out. The following code uniquely illustrates output prepared for presentation.

```plaintext
TITLE1 F=CENTX H=3 'DSGI ABILITIES';
TITLE2 F=CENTX H=1.5 'Points of Interest';
DATA _NULL_;
/*** SET CATALOG WHERE OUTPUT EXISTS ***/
RC=GSET('CATALOG','libref','incat');
RC=GINIT();
RC=GRAPH('CLEAR','SCALE');
RC=GRAPH('UPDATE');
RC=GTERM();
RUN;
```

The output is shown out the next page in figure 6.
CONCLUSION:

This paper has compared two methods available in SAS/GRAPH which have the ability to replay graphs. For PROC GREPLAY the two major distinguishing features would be,

- the windowing feature
- the ability to manage graphic catalogs.

DSGI has enormous capabilities, and the two major features would be,

- the ability to zoom in on a particular part of a graph
- the ability to customize a graph generated by SAS.

These features make the products unique and depending on your needs, will depend on what process is used.

Trademark Notice,

* SAS is a registered trademark of SAS Institute Inc., Cary, NC, USA.

References,

1. SAS/GRAPH, V2, p 1192
2. SAS/GRAPH, V1, p 600
3. SAS/GRAPH, V1, p 604
4. SAS/GRAPH, V1, p 603

Author,

Lynn M. Thorne-Polingo
G.D. Searle
4901 Searle Parkway
Skokie, IL 60077
(708) 982-4872