

DSGI: Some Helpful Hints

Deb Cassidy, Cardinal Health, Inc., Dublin, OH

Introduction

Several presentations have given the advantages and disadvantages to using the DATA Step Graphics Interface (DSGI). In addition to creating custom graphs, advantages include that DSGI can be used with Screen Control Language (SCL), saves disk space, generates graphics faster than the Annotate facility and it supports viewports and windows. One of the disadvantages is that it is one more thing to learn. DSGI can make a graphics person gasp at the sight of GASK while a programmer that thinks in terms of return codes may stumble over drawing a line.

The manual does provide all the information you need but I found myself constantly flipping back and forth through the 130+ pages devoted to DSGI. This presentation was designed as a short summary. You'll still need the manual for reference such as indentifying the different line styles or confirming how default values are selected. I've also provided 'English' terms to help you understand some of the 'computer' terms. The presentation is also generic so I've included a list of articles from Observations and SUGI proceedings that show how to create a graph for specific applications such as multivariate plots or a wind rose.

Getting Started Hints

START SIMPLE! This applies to using SAS/GRAPH® or the Annotate facility as well. Spending a few minutes at the beginning to get acquainted with DSGI can save you hours of frustration later. Your first graph should just use all the defaults for you device and draw the different shapes. Test the same graph on all your devices. Look for things such as round circles and square squares. Are the shapes positioned where you expected? If things are not what you expected, you'll need to investigate what options you might need to change. Changing the number of horizontal/vertical positions and size can be done with a GOPTIONS statement or through DSGI statements. Another option is to have values calculated in your data step based upon your

device. Since there are so many devices (PROC GDEVICE shows over 600 entries) and each site can modify devices, you'll need to determine what is appropriate for your site.

After you have a basic graph, add statements that set attributes such as line types, colors, and sizes. Next try computing the values from the results of GASK routines. You'll also want to read the values from a dataset instead of specifying them directly in the program. Once your comfortable with how DSGI works, it's time for a real graph:

Before you write one line of code, make sure you understand what you are going to graph! Sketch out how you want the graph to look. What shapes do you need? In DSGI, graphs consist of 9 elements.

LINE	a straight line, you can connect several lines together
ARC	a curved line, a circle is an arc of 360 degrees
ELLARC	an elongated curved line, an oval is an elliptical arc of 360 degrees

The next 4 elements differ from the lines above because then can be filled in with a pattern. You can use the first set of elements to draw shapes that look like the second set using an 'empty' pattern. The latter is usually easier, though.

BAR	a rectangle
FILL	a polygon
PIE	a pie wedge
ELLIPSE	an elongated pie wedge

Text can be added with the following elements.

TEXT	regular text
MARK	special characters not found on the keyboard.

You also need to understand how your data are organized and how the data relates to what you want to graph. Many people who have trouble with SAS/GRAPH try to use a dataset that is not set up correctly for that type of graph. For example, do you need one variable and many

observations or do you need fewer observations and several variables. You may need different datasets for different types of graphs even though the graphs are based on the same data. With DSGI, you also need to decide what parts of the graph will be hard-code versus what will be determined by the data. For example, will the axis always be a certain length with tick marks pre-determined or will you calculate the values from your data and the dimensions of your window?

Functions and Routines

DSGI consists of functions and routines. Functions are in the form:

RC=function;

RC stands for Return Code. The statement actually says to perform the requested action and put the return code in a variable named RC. The return code indicates if the action was accomplished so you want a 0. You don't have to use RC as the variable name but it is convenient. If you will be doing some checking on return codes, you'll want to use different variable names. Why would you care to check return codes? For example, you might be concerned with two specific actions. You may have some alternative code to execute if one of them doesn't work. If neither of them work, you want the program to abort and give you the return codes as part of a message. If you use the same variable for all your return codes, you will overwrite the result of the first action.

Routines are in the form:

CALL GASK(attributes to check, RC);

GASK statements provide information about the current status of the system and a RETURN CODE is part of the information. For some routines, the STATUS is the return code so you'll only see one variable in the routine. You can use the information returned from GASK routines as part of your program. When the information to be returned is a character value, you must have already created the variable with an appropriate length statement. Numeric variables will be created for you.

You'll need PUT statements to write the results from the GASK routines to the log. Output statements will write them to a dataset. Don't forget to provide the name of the dataset on the DATA statement. If you use `_NULL_`, you won't get anything because this says to just execute the actions and don't create a new dataset.

The tables at the end of the paper show all the functions and routines. I've grouped them by operating state although I've renamed it the 'Where-can-I-use-it' state.

Index numbers

For some attributes, index numbers are used instead of a description of the attribute. Some are already set by SAS such as fill style. If you have set your shape to be filled with a pattern, a style-index of 2 indicates a cross-hatch of density 2 will be used (Pattern X1). If you specified the shape to be filled with a cross-hatch, the style-index of 2 indicates a cross-hatch of density 1 with a 30-degree angle will be used (Hatch M1X030). For other attributes such as color, you can assign a color to the index.

Quotes or Not?

How do you know whether you need quotes? If you are entering a numeric value for an index number - do not use quotes. If you are entering a variable name - do not use quotes. If you are entering anything else - use quotes. Here are some examples:

```
RC=GSET('FILCOLOR',2);
RC=GSET('FILCOLOR',COLRVAR);
RC=GSET('FILREP',2,1,'HATCH',STYLENO);
RC=GSET('FILREP',4,1,PATRN,3);
LENGTH TYPE $10;
CALL GASK('FILTYPE',TYPE,RC);
```

All the numbers refer to indexes. COLRVAR, STYLENO and PATRN are variable names and the values are either read in from the data or assigned earlier in the program. COLRVAR and STYLENO are numeric variables while PATRN is a character variable. Since fill type is character, I have to create a character variable to retrieve its value. TYPE is a variable name so it does not have quotes.

Bundle or Individual

Bundling enables you to set up a set of attributes and apply the set instead of each individual attribute. You may want to do this because it is easier to draw sections of the graph and jump back and forth between attributes instead of trying to draw all shapes with the same attributes at the same time. Let's say you want to draw two blue, solid-filled circles with a red, cross-hatched circle that is partially on top of one blue circle and partially under the other. (Remember Venn diagrams from high school?) This is a case where you have to draw blue, red, blue.

Another use of bundling is that your data can drive the set of attributes chosen. The following code sets up the blue and red circles. The value of the variable GROUP in your data will determine which set of attributes are used to draw the circle. The data also determines where the circle is drawn. The first three statements tell DSGI to use the attributes as they are bundled. Otherwise, the defaults for the individual attributes will be used and your FILINDEX ignored. Notice there is a style-index specified for the SOLID circle. You must have a value there even though style is ignored for solid or empty circles. For all functions and routines, you must specify all parameters unless they are listed as optional.

```
RC=GSET('ASF','FILCOLOR','BUNDLED');
RC=GSET('ASF','FILSTYLE','BUNDLED');
RC=GSET('ASF','FILTYPE','BUNDLED');

RC=GSET('FILREP',1,3,'SOLID',1);
RC=GSET('FILREP',2,4,'HATCH',2);
(Assumes blue is the third color and red is
the fourth color.)

RC=GSET('FILINDEX',group);
RC=GDRAW('PIE',x,y,30,0,360);
```

Viewports, Windows, and Transformations

Viewports, windows and transformations was probably the most confusing until I tried the following. I created a graph from PROC GSLIDE and then inserted it into a DSGI graph. I tried several different viewports and windows and have included some of the results here. The inside border is the border on the original graph. (Note the graphs have been resized when they were brought into the word-processor.)

```
LIBNAME GRAPHS 'MYID.SASCATLG.GRAPHS';
```

```
PROC GSLIDE NAME='SLD1' BORDER
GOUT=GRAPHS.DSGIDEMO;
TITLE1 H=1 'THIS IS THE TOP';
TITLE2 H=1 A= 90 'THIS IS THE LEFT SIDE';
TITLE3 H=1 A=-90
      'THIS IS THE RIGHT SIDE';
NOTE H=1 ' ' MOVE=(50 PCT, 52 PCT)
      F=MARKER 'H'
      MOVE=(48 PCT, 50 PCT)
      F=MARKER 'F'
      F=CARTOG 'Q'
      F=MARKER 'E'
      MOVE=(50 PCT, 48 PCT)
      F=MARKER 'G';
FOOTNOTE1 H=1 'THIS IS THE BOTTOM';
RUN;
```

```
TITLE;
FOOTNOTE;
DATA _NULL_;
RC=GSET ('CATALOG','GRAPHS','DSGIDEMO');
RC=GINIT();
RC=GRAPH('CLEAR','TEST1');
RC=GRAPH('INSERT','SLD1');
RC=GRAPH('UPDATE');
```

```
RC=GRAPH('CLEAR','TEST2OFF');
RC=GSET ('CLIP','OFF');
RC=GSET ('VIEWPORT',1,.05,.05,.45,.45);
RC=GSET ('TRANSNO',1);
RC=GRAPH('INSERT','SLD1');
RC=GRAPH('UPDATE');
```

```
RC=GRAPH('CLEAR','TEST3OFF');
RC=GSET ('CLIP','OFF');
RC=GSET ('WINDOW',3,10,10,70,70);
RC=GSET ('TRANSNO',3);
RC=GRAPH('INSERT','SLD1');
RC=GRAPH('UPDATE');
```

```
RC=GRAPH('CLEAR','TEST4OFF');
RC=GSET ('CLIP','OFF');
RC=GSET ('VIEWPORT',5,.2,.2,.8,.8);
RC=GSET ('WINDOW',5,10,10,70,70);
RC=GSET ('TRANSNO',5);
RC=GRAPH('INSERT','SLD1');
RC=GRAPH('UPDATE');
```

```
RC=GRAPH('CLEAR','TEST4ON');
RC=GSET ('CLIP','ON');
RC=GSET ('VIEWPORT',5,.2,.2,.8,.8);
RC=GSET ('WINDOW',5,10,10,70,70);
RC=GSET ('TRANSNO',5);
RC=GRAPH('INSERT','SLD1');
RC=GRAPH('UPDATE');
```

```
RC=GRAPH('CLEAR','TEST5OFF');
RC=GSET ('CLIP','OFF');
RC=GSET ('VIEWPORT',6,0,0,1,1);
RC=GSET ('WINDOW',6,-500,-500,200,200);
RC=GSET ('TRANSNO',6);
RC=GRAPH('INSERT','SLD1');
RC=GRAPH('UPDATE');
```

```
RC=GTERM();
RUN;
```

Titles and Footnotes

DSGI recognizes TITLE and FOOTNOTE statements but not NOTE statements. You can accidentally overwrite your titles with your graphs. To avoid this, use the GASK('MAXDISP'..) routine after you have set your titles and footnotes. Use the results to determine the coordinates for the graphic elements. The dimensions returned will exclude any space used by the titles and footnotes.

Example graph

I've included the code to create a simple graph where data from a dataset controls part of the graph. Two graphs are created and then combined as one. I didn't include the result in the paper due to space limitations so you will have to try it and see what you get. Let's hope it puts a smile on your face!

```
LIBNAME GRAPHS 'MYID.SASCATLG.GRAPHS';
DATA ONE;
    EMP=1; MOOD='GOOD'; OUTPUT;
    EMP=2; MOOD='BAD '; OUTPUT;
RUN;

DATA _NULL_;
    SET ONE END=EOF;
    IF _N_=1 THEN DO;
        RC=GSET ('CATALOG', 'GRAPHS', 'DSGIDEMO');
        RC=GINIT();
        RETAIN LLX LLY URX URY YRANGE XRANGE
            RADIUS;
        CALL GASK ('WINDOW', 0, LLX, LLY, URX, URY, RC);
        PUT LLX= LLY= URX= URY=;
        YRANGE=URY-LLY;
        XRANGE=URX-LLX;

        RADIUS=MIN(YRANGE/2, XRANGE/2);
        PUT XRANGE= YRANGE= RADIUS=;
    END;

    RC=GRAPH('CLEAR', 'MOOD1');

    IF MOOD='GOOD' THEN DO;
        STRT=180; END=360; END;
    ELSE IF MOOD='BAD' THEN DO;
        STRT=0 ; END=180; END;
    LENGTH EMPNO $3;
    EMPNO=EMP;

    PUT EMPNO=;

    RC=GDRAW('ARC', XRANGE/2, YRANGE/2, RADIUS, 0, 360);

    RC=GDRAW('ARC', XRANGE/3, YRANGE*2/3, 5, 0, 360);
```

```
RC=GDRAW('ARC', XRANGE*2/3, YRANGE*2/3, 5, 0, 360);
```

```
RC=GDRAW('ELLARC', XRANGE/2, YRANGE/3, 30, 10, 8
    TRT, END, 0);
RC=GSET ('TEXALIGN', 'RIGHT', 'NORMAL');
RC=GDRAW('TEXT', XRANGE/2-
    RADIUS, 5, 'PARTICIPANT #');
RC=GSET ('TEXALIGN', 'LEFT', 'NORMAL');
RC=GDRAW('TEXT', XRANGE/2-RADIUS, 5, EMPNO);
```

```
RC=GRAPH('UPDATE');
IF EOF THEN DO;
    RC=GRAPH('CLEAR', 'MOODALL');
    RC=GSET ('VIEWPORT', 1, 0, .8, .2, 1);
    RC=GSET ('VIEWPORT', 2, .3, .8, .5, 1);
    RC=GSET ('TRANSNO', 1);
    RC=GRAPH('INSERT', 'MOOD1');
    RC=GSET ('TRANSNO', 2);
    RC=GRAPH('INSERT', 'MOOD11');
```

```
RC=GTERM();
END;
RUN;
```

Conclusion

DSGI is a useful tool for creating custom graphics or enhancing existing graphs. Combined with the use of SAS/GRAPH, you can do practically any graph that can be displayed on paper.

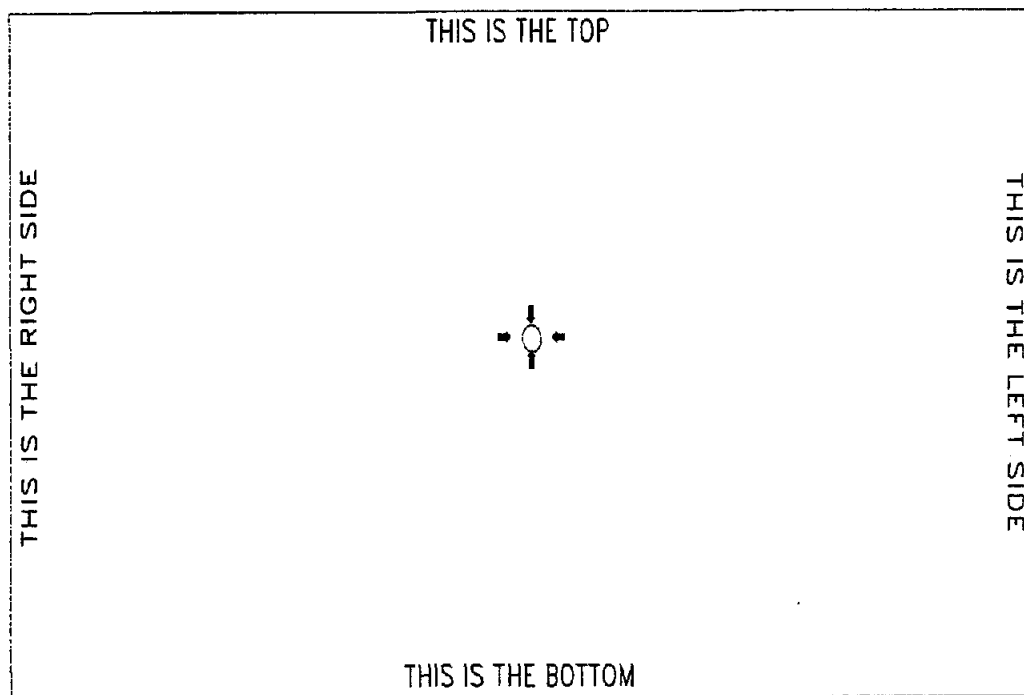
The SUGI presentation will contain some additional detail about the examples that could not fit here. If you would like this additional information, please contact the author.

SAS and the other products are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.

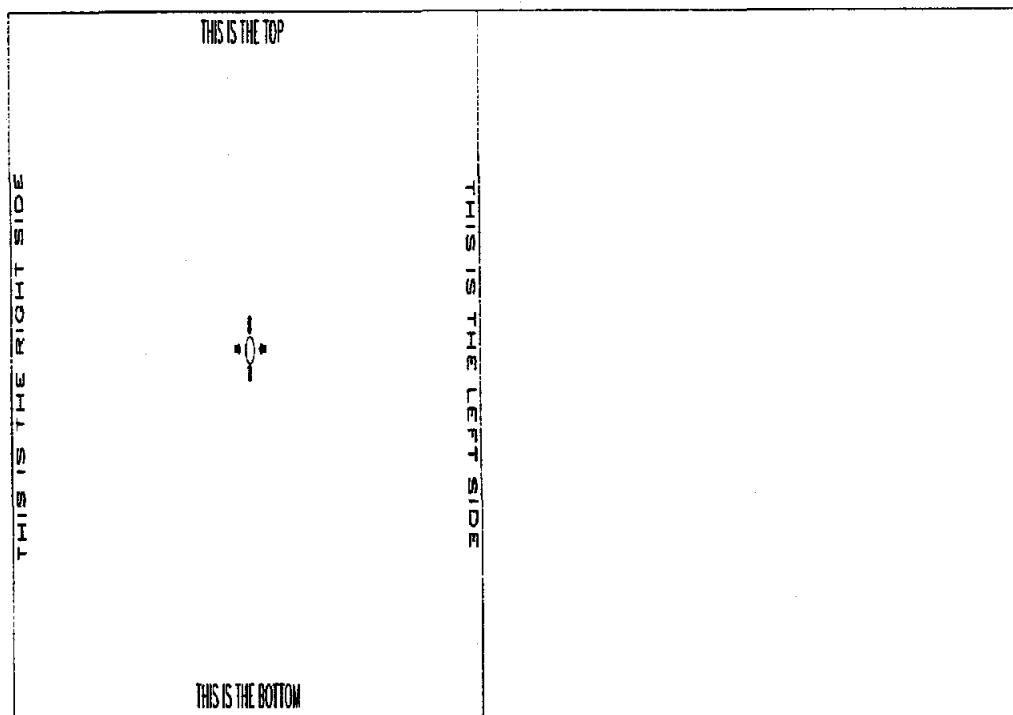
Contact:
(Home)
Deb Cassidy
3809 Secretariat Court
Columbus, OH 43221
(614) 771-7861 (answering machine)

(Work)
Cardinal Health, Inc.
655 Metro Place South, Suite 460
Dublin, OH 43017
(614) 799-6015

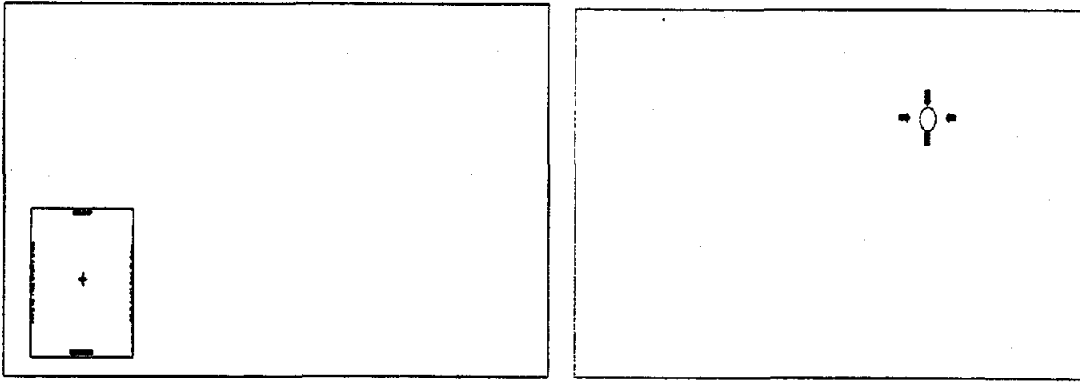
VIEWPORTS, WINDOWS AND TRANSFORMATION EXAMPLES



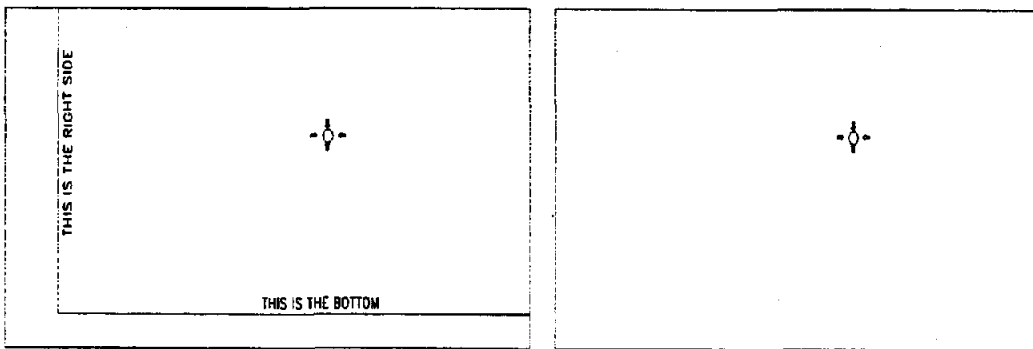
This is the original graph from PROC GSLIDE. It filled the entire screen. (Yes, there is an error in the titles but I found it after I had down-loaded all the graphs. The code in the text is correct.)



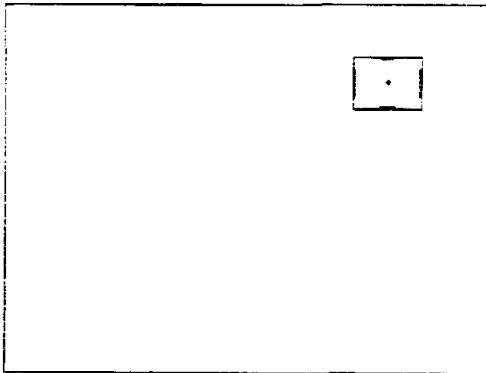
This is the graph as inserted in the default viewport and window for DSGI. Notice it only appeared in the left half of the screen.



The graph on the left is 'TEST2OFF' where the viewport was changed, clip was off, and the window was the default. The graph on the right is 'TEST3OFF' where the window was changed, clip was off, and the viewport was the default.



Both graphs use the same viewport and window which have been changed from the default. The graph on the left has clip set to off while the graph on the right has clip set to on. These are graphs 'TEST4OFF' and 'TEST4ON'.



This graph uses the default viewport but changes the window. This is graph 'TEST5OFF' where I set part of the window coordinates to large negative numbers.

FUNCTIONS AND ROUTINES

Text in upper case indicates text that must be specified as written including the quotes. For functions, lower-case text indicates information you supply either directly as a value or from a variable. For routines, lower-case text are the variables where the status information is returned. Text in italics indicate character variables. Optional parameters are enclosed in <>.

<u>WHERE-CAN-I-USE-IT STATE</u>	<u>FUNCTION</u>
Before DATA _NULL_	If you are using a catalog other than WORK.GSEG, you need to use a LIBNAME statement before you begin your DSGI code. This is in addition to the GSET('CATALOG',...) function.
GKCL - Before GINIT()	Setting Attributes That Affect the Entire Graph GSET('ASPECT',aspect) GSET('CATALOG',libref,catalog-name) GSET('CBACK',cback) GSET('DEVICE',device) GSET('HPOS',hpos) GSET('HSIZE',hsize) GSET('VPOS',vpos) GSET('VSIZE',vsize)
GKCL (Changes to WSAC)	Initializing DSGI GINIT()
WSAC - after GINIT() (Changes to SGOP)	Initializing DSGI for a Specific Graph GRAPH('CLEAR'<,name><,des><,byline>)
SGOP - Between GRAPH('CLEAR') and GRAPH('UPDATE')	Drawing Graphics Elements GDRAW('ARC',x,y,radius,start,end) GDRAW('BAR',x1,y1,x2,y2) GDRAW('ELLARC',x,y,major,minor,start,end,angle) GDRAW('ELLIPSE',x,y,major,minor,start,end,angle) GDRAW('FILL',n,x-values,y-values) GDRAW('LINE',n,x-values,y-values) GDRAW('MARK',n,x-values,y-values) GDRAW('PIE',x,y,radius,start,end) GDRAW('TEXT',x,y,string) Bundling Attributes GSET('ASF',attribute,status) GSET('FILINDEX',index) GSET('LININDEX',index) GSET('MARINDEX',index) GSET('TEXINDEX',index) GSET('COLREP',color-index,color) GSET('FILREP',index,color-index,interior,style-index)

	GSET('LINREP',index,color-index,width,type)
	GSET('MARREP',index,color-index,size,type)
	GSET('TEXREP',index,color-index,font)
	Managing Catalogs
	GRAPH('INSERT',name)
SGOP - Last statement (Changes to WSAC)	Ending DSGI for a Specific Graph
	GRAPH('UPDATE'<,show>)
GKOP,WSOP,WSAC,SGOP - Anywhere between GINIT() and GTERM()	Managing Catalogs
	GRAPH('COPY',name,new-name)
	GRAPH('DELETE',name)
	GRAPH('RENAME',name,new-name)
	Setting Attributes That Affect Graphics Elements
	GSET('COLREP',color-index,color)
	GSET('FILCOLOR',color-index)
	GSET('FILSTYLE',style-index)
	GSET('FILTYPE',interior)
	GSET('LINCOLOR',color-index)
	GSET('LINTYPE',type)
	GSET('LINWIDTH',width)
	GSET('MARCOLOR',color-index)
	GSET('MARSIZE',size)
	GSET('MARTYPE',type)
	GSET('TEXALIGN',halign,valign)
	GSET('TEXCOLOR',color-index)
	GSET('TEXTFONT',font)
	GSET('TEXHEIGHT',height)
	GSET('TEXPATH',path)
	GSET('TEXUP',upx,upy)
	Defining/Activating Viewports & Windows
	GSET('CLIP',status)
	GSET('VIEWPORT',n,llx,lly,urx,ury)
	GSET('WINDOW',n,llx,lly,urx,ury)
	GSET('TRANSNO',n)
WSAC - (Changes to GKCL)	Ending DSGI
	GTERM()
All operating states	Handling Message
	GDRAW('MESSAGE',message)
	GSET('MESSAGE',status)
	GPRINT(code)

<u>WHERE-CAN-I-USE-IT STATE</u>	<u>ROUTINE</u>
SGOP - Between GRAPH('CLEAR') and GRAPH('UPDATE')	Checking Attribute Settings GASK('COLINDEX',n,index-array,rc) GASK('COLREP',color-index,color,rc) Querying Catalogs GASK('OPENGRAPH,name,rc) Checking Attribute Settings GASK('COLINDEX',n,index-array,rc) GASK('COLREP',color-index,color,rc)
GKOP,WSOP,WSAC,SGOP - Anywhere between GINIT() and GTERM()	Checking Attribute Bundles GASK('ASF',attribute,status,rc) GASK('FILINDEX',index,rc) GASK('LININDEX',index,rc) GASK('MARINDEX',index,rc) GASK('TEXINDEX',index,rc) GASK('FILREP',index,color-index,interior,style-index,rc) GASK('LINREP',index,color-index,width,type,rc) GASK('MARREP',index,color-index,size,type,rc) GASK('TEXREP',index,color-index,font,rc) Checking Attribute Settings GASK('FILCOLOR,color-index,rc) GASK('FILSTYLE',style-index,rc) GASK('FILTYPE',interior,rc) GASK('LINCOLOR',color-index,rc) GASK('LINTYPE',type,rc) GASK('LINWIDTH',width,rc) GASK('MARCOLOR',color-index,rc) GASK('MARSIZE',size,rc) GASK('MARTYPE',type,rc) GASK('TEXALIGN',halign,valign,rc) GASK('TEXCOLOR',color-index,rc) GASK('TEXEXTENT',x,y,string,x-end,y-end,x1,x2,x3,x4,y1,y2,y3,y4,rc) GASK('TEXFONT',font,rc) GASK('TEXHEIGHT',height,rc) GASK('TEXPATH',path,rc) GASK('TEXUP',up-x,up-y,rc) Checking Attributes That Affect the Entire Graph GASK('ASPECT',aspect,rc)

GASK('CATALOG',libref,memname,rc)
 GASK('CBACK',cback,rc)
 GASK('DEVICE',device,rc)
 GASK('HPOS',hpos,rc)
 GASK('HSIZE',hsize,rc)
 GASK('MAXDISP',units,x-dim,y-dim,x-pixels,y-pixels,rc)
 GASK('VPOS',vpos,rc)
 GASK('VSIZE',vsize,rc)

Querying Catalogs

GASK('GRAPHLIST',n,name-array,rc)
 GASK('NUMGRAPH',n,rc)

Checking Transformation /Viewport/ Window Definitions

GASK('TRANS',n,vllx,vlly,vurx,vury,wllx,wlly,wurx,wury,rc)
 GASK('TRANSNO',n,rc)
 GASK('CLIP',status)
 GASK('VIEWPORT',n,llx,lly,urx,ury,rc)
 GASK('WINDOW',n,llx,lly,urx,ury,rc)

All operating states

Checking system status

GASK('STATE',status)
 GASK('WSACTIVE',status)
 GASK('WSOPEN',status)

OTHER DSGI SOURCES

Observations:® The Technical Journal for SAS Software Users

- 1Q92 Picture This section
- 4Q92 Drawing a Pictogram: An Adventure into the DATA Step Graphics Interface, Walker and Jeffreys
- 2Q94 Creating Multivariate Plots Using the DATA Step Graphics Interface, Walker and Olszewski
- 3Q94 Picture This section
- 1Q95 Input/Output section

Proceedings of MWSUG

- 1991 A Comparison of PROC GREPLAY and the DATA Step Graphics Interface (DSGI), Thome-Polingo
- 1994 DSGI: The OTHER Way to do SAS Graphics, Whitney

SUGI Proceedings

- 14 The DATA Step Graphics Interface, Walker

SAS Manual

SAS/GRAPH Software: Reference, Version 6, First Edition, Volume 1, Chapters 20 & 21