

# SAS/AF<sup>®</sup> Interface For Dynamic Hotspotting of SAS/GRAPH<sup>®</sup> Outputs

*Robert Peszek and Izabella Peszek*

Robert Peszek works as a Clinical Programmer/Analyst for PPD-Pharmaco Inc. His current areas of expertise and interests are SAS programming, databases, and software development using PowerBuilder and Java. He has a PhD in Applied Mathematics from the University of Maryland and is a Certified PowerBuilder Developer. Robert has been using SAS software for 4 years.

Iza Peszek works as a biometrician at Merck and Co. Previously, she worked as a senior statistician for Ohmeda, Inc. A SAS user for 8 years, her interests include statistical programming, graphical presentation of clinical data, and automation of the report writing. Iza holds a PhD in Statistics from the University of Maryland and M.S. in Applied Mathematics from the University of Wroclaw, Poland. Robert and Iza have been married for 11 years.

## Abstract

This article presents a SAS/AF<sup>®</sup> FRAME Entry application that allows dynamic hotspotting of existing SAS/GRAPH<sup>®</sup> plots. It allows users to click on points on the graph, which in turn executes SAS code to produce output in the form of graphs, tables, and listings.

## 1. Introduction

This article presents a simple SAS/AF application allowing dynamic hotspotting of any SAS/GRAPH output generated with PROC GLOT. We will refer to this program as the “Interactive Graph” application. This application allows users to click on points on the graph and view various types of information about the selected point in the form of graphs, tables, and listings. The application was developed and tested on SAS for Windows, Releases 6.11 and 6.12.

The idea for this article originated from the needs of the pharmaceutical industry and was prompted by the frequent requests from the regulatory reviewers for interactively displayed graphs of adverse events as a part of CANDA (Computer Assisted New Drug Application). However, this application could be used in many other business environments. Our idea was to create an AF application that can be used by SAS programmers not familiar with SAS/AF. Once this application is created (or copied from this article), it can be invoked at the end of the existing SAS programs generating plots or can be used with graphs stored as catalog entries. Typically, only a few lines of code are needed to invoke the application and to specify the desired interactive features. All this programming is reduced to SAS/GRAPH and base SAS and is very similar to programming with the SAS/GRAPH Annotate Facility. Such an approach makes interactive graphing available to companies employing few or no SAS/AF developers.

To emphasize the similarity between our approach and the SAS/GRAPH Annotate Facility, and the fact that the presented application **does not** employ the SAS/AF Hotspot class, we will use the term “interactive annotation” instead of “hotspotting”.

There are many ways to add interactive functionality to graphic outputs in SAS, for example, using SAS/INSIGHT<sup>®</sup>, DSGI, or SAS/AF. In SAS/INSIGHT, users can click on the graph and get limited information about the selected point. DSGI can cooperate with SCL programs, which can be used for creating interactive graphs. Within SAS/AF, interactive functionality can be obtained with SAS/GRAPH output, Hotspot, and Graphics classes. SAS/AF software allows you to create hotspots in the SAS/GRAPH output object. The user can click on a hotspot to trigger the execution of different sections of code.

Unfortunately, hotspots are static objects in the sense that they are linked to a particular SAS/GRAPH catalog entry. The Graphics object class provides dynamic interactive functionality, allowing users to select a part of the graph or a point on a scatterplot. However, Graphics objects can generate only certain types of graphs, which is why we developed this new hotspotting method.

We present an alternative way to create hotspots for interactive annotation of SAS/GRAPH output. Our idea is to join the graph-generating power of SAS/GRAPH with the interactive capabilities of the Graphics class. A SAS/GRAPH Output object is used to display the plot only, while an overlapping transparent Graphics object, aligned with the plot, forms a top layer of interactive points.

Section 2 gives an example of an Interactive Graph session and explains interactive features available to users. Section 3 describes base SAS and SAS/GRAPH programming required to interactively annotate SAS/GRAPH output. A simple program presented in Section 4 can be used by a SAS programmer as a learning tool or by a SAS/AF developer to make a proper alignment of objects in the Interactive Graph window. Section 5 covers the details of AF and SCL programming, and describes how to align the SAS/GRAPH Output and Graphics objects. Such alignment needs to be performed each time the display resolution settings are changed.

Possible improvements, such as nesting interactive graphs or modifying programming techniques to comply with Object Oriented programming principles, are discussed in Section 6. The coordinates of the interactive points are expressed as a percentage of the graph output area. Section 7 presents an example calculation of these coordinates.

Users of this application (not necessarily familiar with SAS ) should read Section 2. Programmers who intend to use a developed AF component with their programs should read Sections 2, 3, 4 and 7. Sections 5 and 6 are for SAS/AF developers who would like to understand our application or customize it. The reader not familiar with SAS/AF and SCL can skip these sections (although only basic knowledge of AF and SCL is required to read Section 5).

## 2. How to Use Interactive Features

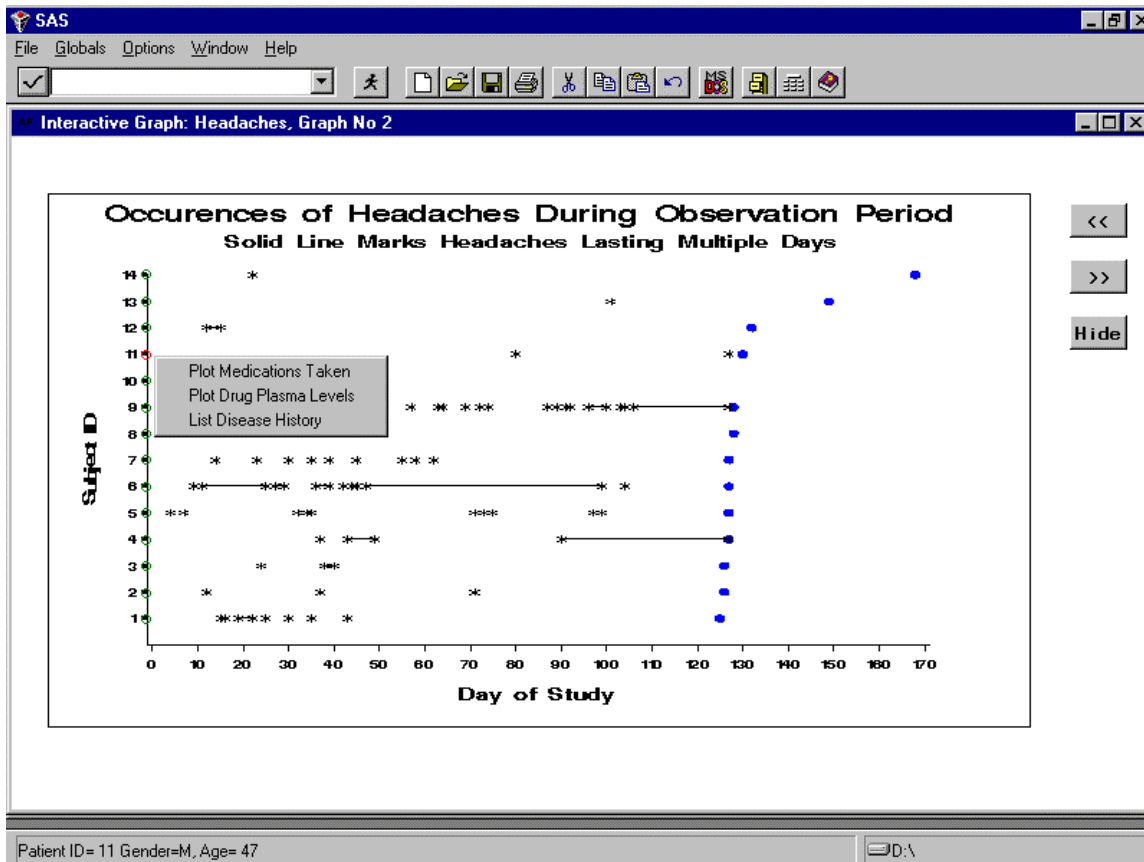
This section explains how the Interactive Graph application works from the user's point of view. The Interactive Graph application displays the window shown in Figure 1.

This window contains a graph output area and three Push Buttons. The information about the currently displayed graph and the selected point is displayed in the window's title bar and in the message line located at the bottom of the SAS window.

The graph output area displays a graph produced with SAS/GRAPH PROC GPLOT. Some of the points on this graph have interactive functionality. Clicking on such a point displays a short description of the selected point in the message line. Right-clicking on the point displays a pop-up menu of available actions (submittable SAS programs). The menus may be different for different points on the graph. Clicking on an item in the pop-up menu passes the information about the selected point to an appropriate SAS program and submits it. In this way, the user may run SAS programs that display additional information about the selected point in the form of graphs, tables, and listings.

The Show/Hide Push Button can be used to view all points with interactive functionality (in Figure 1, all interactive points are located on the vertical axis). Clicking on this button highlights all interactive points as green circles. This button acts as a toggle switch—clicking on it again renders all interactive points invisible. Selecting an interactive point in the “show” mode displays a red circle and has the same effect as in previous paragraph.

“Previous” (<<) and “Next” (>>) buttons display other available graphs if more than one SAS/GRAPH output is linked to an interactive annotation. The “Previous” button is hidden (is not visible and not active) if the first graph is displayed, while the “Next” button is hidden for the last graph on the list.



**Figure 1.** A snapshot of the interactive session. The graph presents adverse events for 14 patients (mock-up data). All interactive points are located on the vertical axis and correspond to patient numbers. The message line displays a short description of the selected point, and the pop-up menu lists the available Menu Execution Programs.

### 3. How to Create Interactive Annotation

Setting up the application with a new graph output requires only SAS/GRAPH and base SAS programming. We will call such programming “the interactive annotation of the graph”. This section explains how to prepare this interactive annotation.

The process is very similar to using the Annotate Facility with SAS/GRAPH software. The difference is that the interactive annotation is done after the graph has been created and stored as a catalog entry. Similarly to the SAS/GRAPH Annotate Facility, we need to create an annotate data set (referred to as an IANNO data set). The structure of an IANNO data set resembles that of a simple annotate data set. This data set contains the information about the coordinates of the interactive points and the specifications of the interactive functions that can be performed when these points are selected at run time.

Two programming steps are required to use the Interactive Graph Application. The first is to write a SAS Parent Program that creates an IANNO data set and specifies other details of the interactive annotation (for example, which SAS/GRAPH catalog entry is used). These details are specified in the data set WORK.G\_LIST. The second step is to write SAS programs, called Menu Execution Programs, that will be submitted when the user clicks on a pop-up menu item of a selected point. A simple example is given in the next section. For clarity, we have divided the SAS programming required to set up the Interactive Graph Application into the following small steps, explained below:

- Step 1. (Parent Program) Include the program generating the SAS/GRAPH output.
- Step 2. (Parent Program) Create the G\_LIST data set.
- Step 3. (Parent Program) Create the IANNO data sets.
- Step 4. (Parent Program) Invoke the AF application.
- Step 5. (Menu Execution Programs) Write the Menu Execution Programs.

Step 1. The program generating the graphic output needs to be included in the Parent Program. You need to identify the full name of the catalog entry containing the graph. It is a good idea to use the GOUT option in PROC GPLOT. If the graph output is stored in a permanent catalog, this step may be omitted, but you need to specify the libname of the catalog before the AF application is invoked.

The interactive annotation will need the coordinates of each interactive point expressed as the percentage of the SAS/GRAPH output area. You may need to adjust GOPTIONS, LABEL, and AXIS statements in the included program to facilitate the calculation of these coordinates. Please refer to Section 7 and to the details of Step 3 for more information.

Step 2. The data set G\_LIST, created in the WORK directory, stores information on how to link a SAS/GRAPH output with IANNO data sets. It contains the following variables:

GOUT_NM	\$40	Specifies the complete address of the graph output. GOUT_NM = <i>'libref.catalog-name.entry-name'</i> , where <i>entry-name</i> is one of the following: <i>gplot</i> or <i>gplot1</i> , <i>gplot2</i> , .... For example, if no permanent catalog is used then GOUT_NM = <i>'gseg.gplot'</i> for the first graph created (both permanent catalogs and default temporary catalogs can be used).
IANNO	\$40	Specifies the data set used for interactive annotation of the graph output. IANNO = <i>'libref.dataset-name'</i> . Step 3 explains how to define IANNO data sets.
G_LABEL	\$40	The title displayed on the window's title bar. The default title is "Interactive Graph". If G_LABEL is specified, the title will display "Interactive Graph: <i>value-of-G_LABEL</i> ".

Each observation in G\_LIST corresponds to one SAS/GRAPH output. The Interactive Graph application can be used to interactively annotate more than one graph output at the same time. In the simplest case, when you use only one graph, the G\_LIST data set has one observation.

Step 3. The IANNO data set specified in G\_LIST should contain the following variables:

XVAL	numeric	Variables XVAL, YVAL specify the coordinates of the interactive point on the graph (as the percentage of the SAS/GRAPH output area).
YVAL		
MARKER	numeric	Presently, MARKER=1 for all points. In future releases, this variable will be used to customize the appearance of different groups of interactive points.
PNT_ID	\$40	Contains information about the interactive point. This information will be passed to the Menu Execution Programs.
DESC_ST	\$40	A short description of the interactive point that will appear on the message line when the point is selected.

EXE1	\$40	(optional) The full pathname of the first Menu Execution Program.
NAME_E1	\$40	(only if EXE1 is used) The pop-up menu name for the first Menu Execution Program.
...	...	...
EXE9	\$40	(optional). Note: The current version of the program limits the number of the Menu Execution Programs to 9 (see for example the LENGTH statement in the INIT block of the SCL program in the Appendix); a simple modification of the SCL program would allow for more than nine.
NAME_E9	\$40	(only if EXE9 is used).

Each observation in the IANNO data set will define one interactive point on the graph. The location of the point is specified using the variables XVAL and YVAL and is expressed as the percentage of the SAS/GRAPH output area (similarly to X and Y coordinates corresponding to XSYS = '3' and YSYS='3' in the SAS/GRAPH Annotate Facility). In Section 7, we give an example calculation of the XVAL and YVAL coordinates.

Once the user selects a point in the SAS/GRAPH output, our program will show a pop-up menu of the available actions. Each action corresponds to one SAS program (Menu Execution Program). Selecting an item from this menu will pass the value of the character variable PNT\_ID to the appropriate Menu Execution Program and will submit the program.

Defining the values of PNT\_ID depends on what interactive functionality you want. The following example illustrates this. Consider a plot of blood pressure over time for several patients. One additional piece of information you want to display is the patient's age and gender. PNT\_ID value can be equal to the patient's identification number. The Menu Execution Program will access the data set containing age and gender information, and it will print the record with the specified patient ID. On the other hand, if you want to know what medications this patient took on the day of the blood pressure reading, PNT\_ID has to include the patient's ID and the value of the time variable, for example, PNT\_ID="if PAT\_ID=10 and DAY=3". Note: PNT\_ID is simply a 'string' macro variable that is available for use inside the Menu Execution Programs.

The variables EXE1—EXE9 contain the names of the SAS programs (Menu Execution Programs) that correspond to the pop-up menu items. The names of the menu items are defined by the NAME\_E1, NAME\_E2, ..., NAME\_E9 variables. Selecting the *i*<sup>th</sup> item from the menu submits the following line:

```
%include value_of_EXEi;
```

(A clever idea is to specify a FILENAME statement in the Parent Program for the directory containing the Menu Execution Programs and to specify EXEi in the form EXEi = 'fileref(name\_of\_interactive\_program)' ).

In addition, the following global macro variables are passed to the EXEi programs and can be used to restrict the retrieval of information: &desc\_st, &graph\_no, &pnt\_id, &menu\_no. Refer to Step 5 for more information on these variables.

Selecting a point on the graph displays the value of the variable DESC\_ST in the message line. This is used for a short description of the selected point.

**Step 4. Include the following statements:**

```
LIBNAME INTGRAPH 'directory containing the catalog with AF application';
DM 'AF C=' INTGRAPH.name_of_catalog.IGRAPH.FRAME' continue;
where name_of_catalog is the name of the catalog containing the AF application.
```

Step 5. The final step is to provide all needed Menu Execution Programs. The AF application will invoke a Menu Execution Program by submitting the following statement:

```
%include value_of_EXEi;
```

The AF application will define the following global macro variables (which can be used in the Menu Execution Programs)

- &GRAPH\_NO (= 1, 2, ...) specifies which graph the user is viewing and is equal to the observation number in the G\_LIST data set. This is useful if the Menu Execution Program is written for more than one graph. It is also convenient in nested uses of the AF application (see Section 6).
- &PNT\_ID identifies which point has been selected. This information is provided by the PNT\_ID variable in the IANNO data set (see Step 3).
- &MENU\_NO (= 1, 2, ...) identifies which item on the menu has been selected. This is convenient if the same Menu Execution Program is used for more than one item on the pop-up menu. For example, the IANNO variables EXE1 and EXE2 may have the same values (refer to the same program) but selecting NAME\_E1 and NAME\_E2 may invoke different parts of this program.
- &DESC\_ST is a short description of the selected point provided by DESC\_ST variable in the IANNO data set. You may use it in the output titles inside the Menu Execution Programs

Please refer to the next section for an example.

## 4. Example of Interactive Graph Annotation

The following example shows how the interactive annotation works, and it will help to design and test the FRAME entry. This program produces a simple scatterplot with four points, the G\_LIST data set, and the IANNO data set. These data sets and the plot will allow easy alignment of the SAS/GRAPH output object and the Graphics object. Use this program to build the application, to realign interactive points with the graph each time resolution settings are changed, and as a training tool to learn how to set up the Parent Program and the Menu Execution Programs.

Note for SAS/AF developers: The outputs of this test program are referenced in the AF application. If you decide to delete them, make sure to remove appropriate attributes from the SAS/GRAPH Output and the Graphics objects in the AF FRAME entry.

```
/*-- test Parent Program --*/
libname intgraph 'directory_where_AF_application_will_be_stored';
filename grph_prg 'directory_where_test_menu_execution_program_will_be_stored';

/*-----define data set to produce graph -----*/
/*-- variable POINT identifies the point on the graph and --*/
/*-- will be used later to define PNT_ID and DESC_ST --*/
/*-- x and y are used to produce the plot y*x and define XVAL, YVAL --*/
data intgraph.test;
input x y point;
cards;
0 0 1
1 8 2
8 1 3
10 10 4
;
run;
```

```

/*-----produce graph -----*/
/*-----this graph will be 'interactively annotated' ---*/
GOPTIONS
    device=win
    rotate=landscape
    vorigin=0 in
    hororigin=0 in
    vsize=0 in
    hsize=0 in
    gunit=pct
    ;
AXIS1  length= 70 pct
       label= none
       minor=none
       origin=(20 pct, 20 pct) offset=(0,0) order=0 to 10;
AXIS2  length= 70 pct
       label= none
       value=(f=swiss h=2)
       minor=none offset=(0,0) order= 0 to 10;

proc gplot data= intgraph.test gout=intgraph.testgseg;
plot y * x/
      HAXIS=AXIS1
      VAXIS=AXIS2;
run;

/*-----define G_LIST data set-----*/
data g_list;
gout_nm='INTGRAPH.TESTGSEG.GPLOT'; ** specify where the plot is located **;
ianno='INTGRAPH.ANNOTEST'; ** specify location of IANNO data set for the plot**;
g_label='TEST GRAPH'; ** specify title bar**;
run;

/*-----define test IANNO data set-----*/
data twopoint;

/*these two points mark bottom left and top right, needed to do alignment*/
xval = 0; yval = 0; output;
xval = 100; yval = 100; output;
run;

data intgraph.annotest;
set twopoint intgraph.test (in=ds2);
if ds2 then do;
xval = 20+ 70* x/10; ** specify location of interactive points **;
yval = 20 + 70* y/10; ** as percent of the output area **;
** see LENGTH, ORIGIN in def of AXIS1,AXIS2 **;
desc_st = 'POINT NO: '||put(POINT,1.); ** description to be displayed **;
**in message line if this point **;
** is selected **;
exe1 = 'GRPH_PRG(EXE1)'; ** location of SAS program to be executed if **;
** first item selected on the menu **;
name_e1 = 'SHOW GRAPH'; ** text displayed on first menu item **;
exe2 = 'GRPH_PRG(EXE1)'; ** location of SAS program to be executed if **;
** second item selected on the menu **;
name_e2 = 'SHOW LISTING'; ** text displayed on second menu item **;
pnt_id = 'if point='||PUT(POINT,1.); ** info passed to SAS pgm (MEP) **;
end;
marker =1;
run;

/*--set up a file used as input in the Menu Execution Program -----*/
/*--the variables x2 and y2 are additional information (to be listed ---*/
/*--or plotted by Menu execution program) for the user selected values ---*/
/*--of 'POINT' ---*/
data intgraph.details;
input point x2 y2;
cards;
1 -1 4
1 0 1
1 1 3
1 2 4
2 -1 4
2 0 2
2 1 3
2 2 4

```

```

3 -1 4
3 0 3
3 1 3
3 2 4
4 -1 4
4 0 4
4 1 3
4 2 4
;
run;

/*----- to build the application run this-----*/
* proc build c=intgraph.afgraph; run;

/*----- to test the application run this-----*/
* dm 'af c=intgraph.afgraph.igraph.frame' continue;

```

Save the above Parent Program (filename is not relevant).

To build the frame and to make a test run, we need the Menu Execution Program, displayed below.

```

**** test Menu Execution Program ****;

****-- GET DATA FOR THE SELECTED POINT---;
data subgraph;
set intgraph.details;
&pnt_id; ** 'if statement' passed to the program (PNT_ID defined in IANNO data set)**;
run;

****-- note MENU_NO and DESC_ST are passed to the program, MENU_NO tells      **;
****-- the program which menu item was selected by the user; DESC_ST is a short **;
****-- description of selected point and is defined in IANNO data set        **;
%macro OUTPUT;
%if &menu_no = 1 %then %do;

****--FIRST MENU ITEM SELECTED-----*;
title "Graph for &desc_st";
proc gplot data=subgraph;
PLOT y2 * x2;
run;
%end;
%else %if &menu_no =2 %then %do;

****--SECOND MENU ITEM SELECTED-----*;
title "Listing for &desc_st";
options nocenter;
proc print; run;
%end;
%mend OUTPUT;

%OUTPUT;

```

Save this program as EXE1.SAS in a folder specified in the FILENAME statement (the third line of the Parent Program). If you are not interested in AF development process and already have the SAS catalog file containing the SAS/AF application, you can now do a test run. Note: You may need to adjust the alignment of the “igraph.frame” entry (contained in the catalog file AFGGRAPH.SC2) to match the display resolution settings of your computer. The alignment process is explained in next section. To do that, make sure that the catalog containing the application is located in the directory specified in the LIBNAME statement (the second line of the Parent Program) and that it is named AFGGRAPH.SC2. Uncomment the last line of the Parent Program:

```
dm 'af c=intgraph.afgraph.igraph.frame' continue;
```

Then run the whole Parent Program.

## 5. Building the SAS/AF Application

This section is intended for SAS/AF developers. The reader should be familiar with the basics of the SAS/AF FRAME Entry and SCL (see the References section).

We will use a SAS/GRAPH output object to display the graph. The interactive functionality comes from overlapping a transparent Graphics object. We will need to align the SAS/GRAPH and the Graphics objects as described below. Unfortunately, if the application is used with a display resolution setting different from the one used for the alignment, the points may appear slightly misaligned (the application was originally developed using 800x600 resolution). We tested our application on several display settings and found that the distortion was minimal. Still, if you anticipate that the application will be used with various display settings, you may want to design a separate FRAME entry for each resolution.

This section describes how to build the FRAME entry from scratch. It will provide useful information even if you are working with existing catalog containing the FRAME entry.

Before getting started, make sure to run the Parent Program from the previous section with the PROC BUILD step uncommented. Make sure that you have modified lines 2 and 3 of this program and that the Menu Execution Program EXE1.SAS is in place. Running PROC BUILD opens a new window listing the contents of AFGGRAPH catalog. Edit the frame IGRAPH.FRAME, and modify the title in the General Attributes window to 'Interactive Graph'.

### Step 1

“Make” the following transparent objects: Container Box, SAS/GRAPH Output, and Graphics. The Graphics object should overlap the SAS/GRAPH output as shown in Figure 2 (it is important that it does not overlap the Container Box).

Set the following attributes for the SAS/GRAPH Output object:

- Name: GRAPH
- Graph: INTGRAPH.TESTGSEG.GPLOT.GRSEG
- No scrollbars
- Graph Size: Resize to fit
- Set Region Attributes: Outline Type to Simple.

Set the following attributes for the Graphics object:

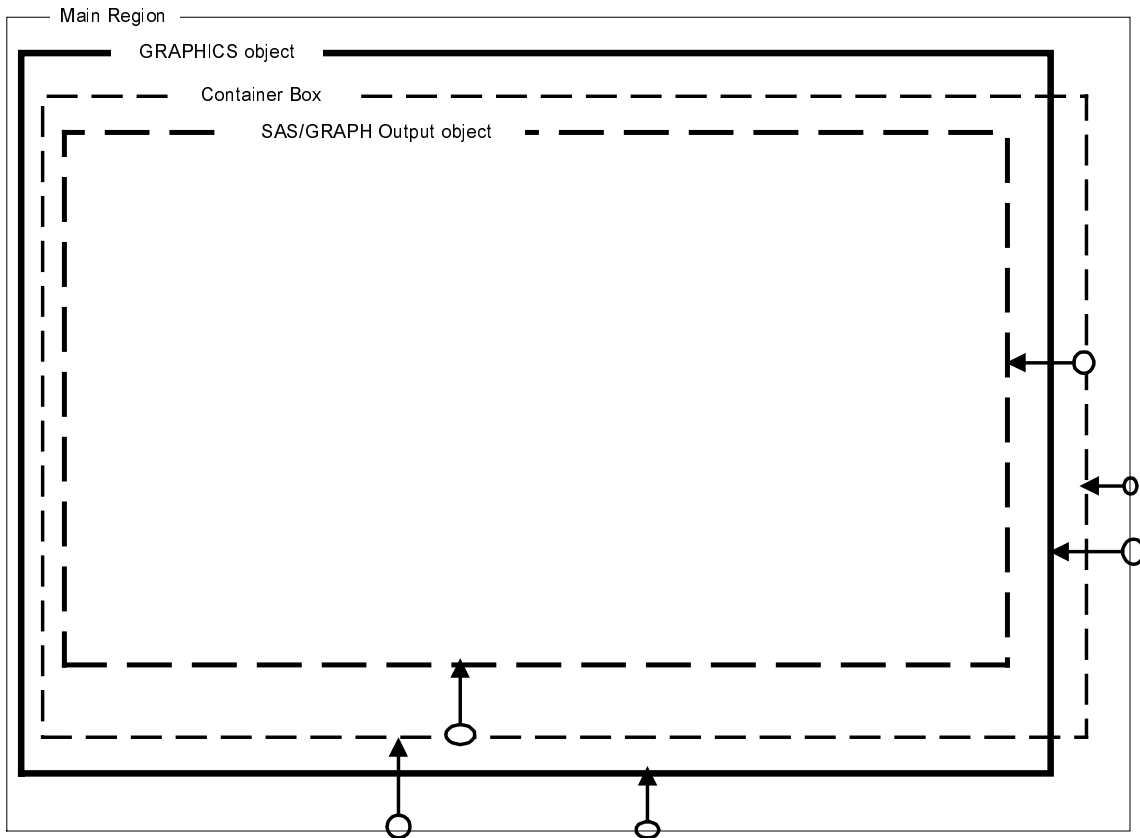
- Name: GRAPHIC
- Graph Type: Scatter
- Data Specifications:
  - Data set name: INTGRAPH.ANNOTEST
  - X Variable: XVAL
  - Y Variable: YVAL
  - ID Variable: MARKER
  - Data Source: Data set
  - Sort Order: NONE
- Legend: not selected
- Modify Appearance from Additional Attributes list by selecting white color for text (same as background) and smallest Fonts on both AXES (such as Courier New size 1).
- Modify Command Processing Pop Menu settings to: Run Object Label.
- Turn “data info off” and “highlight on” by right-clicking on the object and selecting the appropriate items from the pop-up menu.

### Step 2

In this step, we describe how to align SAS/GRAPH Output and Graphics objects. In SAS/AF, a graphical object can partially or entirely overlap other graphical objects. Partially overlapping objects are called “siblings”. A “parent” is an object that entirely overlaps another object (its child). Child

objects cannot have focus or be selected by user. On the other hand, sibling objects can be placed in the foreground or in the background during design by right-clicking on the sibling object and choosing Push (to background) or Pop (to foreground) from the menu. Only the object Popped to foreground can be selected by the user.

For this reason, it is important to place all the objects as shown in Figure 2, with the Graphics object Popped to the foreground. The Container Box is a parent to the SAS/GRAPH object (overlaps the SAS/GRAPH Output object) but is a sibling to the Graphics object. Because the Graphics object is a sibling to the Container Box, it is also a sibling to any child of the Container Box. Thus, the Graphics object and the SAS/GRAPH Output object are siblings and each of them can be placed in the foreground. Make sure you leave enough space to the right of Container Box for the three Push Buttons shown on Figure 1. It is best if the Push Buttons do not overlap with the objects outlined in Figure 2.



**Figure 2.** Placement of objects

After the attributes are correctly entered, the Graphics object should display a scatterplot in a rectangular area with 10 tick marks on each axis. At this time, these points may not be aligned with points on the SAS/GRAPH Output object. If you are working in SAS 6.12, the SAS/GRAPH Output object may not be visible at this point. The points in the IANNO data set range from (0, 0) to (100, 100) and reference the percentage of the SAS/GRAPH Output display area.

In SAS 6.11: If the SAS/GRAPH Output object is not selected, its outline appears as a thin dashed line. Resize the Graphics object (it needs to be Popped to the foreground first) so that the bottom-left and the top-right corners of the SAS/GRAPH Output object are perfectly aligned with the bottom-left and the top-right “+” marks corresponding to the (0, 0) and the (100, 100) points. After this is done, all other points should be aligned.

In SAS 6.12: The Graphics object displays at the design time with a nontransparent gray background (“backplane”). This makes the alignment procedure a little more complicated than in SAS 6.11. To make the proper alignment, follow the following steps:

- Pop the Container Box by right-clicking on it in the area not covered by the Graphics object (on the right ) and choosing Pop.
- Left-click in the middle of the Container Box, which will reveal a semitransparent frame of the SAS/GRAPH Output object. Within this frame you should see the reference “+” marks and the scatterplot. If you cannot see the scatterplot, do one of the following: ALT-TAB to a different application and ALT-TAB back, or right click on the SAS/GRAPH Output object frame, choose Object Attributes from the menu, and select “Cancel”. Either method refreshes the display, and the plot becomes visible.
- Now you are ready to resize the SAS/GRAPH Output object. Move the corners of the object frame so that they overlap with the bottom-left and the top-right “+” marks corresponding to the (0, 0) and the (100, 100) points. If you have not modified color settings , the frame and plot will appear in black, while the “+” scattered points will appear in color. After aligning the corners, all other points should be aligned also.
- Right-click on the Container Box outside the SAS/GRAPH Output object and choose Push. This places the Graphics object in the foreground. The scatterplot is no longer visible, but the gray background will be made transparent programmatically in the SCL program.

We can enhance the application by allowing the user to change the size of the graph by resizing the window. This step is optional, but if you skip it, an attempt to resize the window may make a part of the graph and/or the Push Buttons disappear. To allow run-time resizing, define the absolute directional attachments as shown in Figure 2 (See *SAS/AF Software: FRAME Entry Usage and Reference, Version 6, First Edition*, pp. 48-58, and *FRAME Application Development Concepts, Version 6, First Edition*, pp. 21-30, for details on defining attachments). Specifying such attachments guarantees that the constant distance is kept from the right and bottom sides of the window when the window is being resized. Because the offsets between graphic display area and the border of the Graphics Object are constant, resizing does not cause misalignment. Defining attachments this way will still allow you to resize the Graphics object and the SAS/GRAPH Output object during design, and assures you the objects stay aligned when the window is resized.

### Step 3

Create the Push Buttons and assign the following attributes to them respectively:

- Name: PREV\_PB, label: <<
- Name: NEXT\_PB, label: >>
- Name: SHOW\_PB, label: Show.

### Step 4

It is a good idea to keep Push Buttons away from the container object and the graphics object so they will not become accidentally hidden in the background when other objects are being Pushed and Popped. If it happens, make sure that the Graphics object and the Push Buttons are all in the foreground by right-clicking on them and selecting ‘POP’ from the menu).

### Step 5

Edit the IGRAPH.SCL entry, and include the program given in the Appendix.

Note that the Menu Execution Programs cannot be submitted with TESTAF. To fully test the application, you need to run it as described in Step 4 of Section 3.

## 6. Possible Improvements

You are welcome to modify the presented Interactive Graph application. The SCL program was written without much regard for the principles of the Object Oriented Programming because we wanted it to be approachable to SAS programmers not familiar with OOP techniques.

In addition to making some use of the Object Oriented Technology, the application can benefit from some small enhancements. The SUBMIT block (refer to the code in the Appendix) is responsible for executing the Menu Execution Program selected by the user and passing information to this program. Assume that we have replaced this block with the following:

```
submit:
  link close_ds;          ** close data sets so MEP can modify them **;
  submit continue;
  GOPTIONS NODISPLAY;
  %let click_pt = &submt;  ** pass the parameters to the MEP **;
  %let par_dsc = &desc_st;
  %let graph_no = &graph_no;
  %let menu_no = &p_selec;
  %include &progexe;      ** run MEP **;
  GOPTIONS DISPLAY;
  endsubmit;
  graph_no = &graph_no;  ** redefine the graph number to be displayed **;
  link open_ds;          ** open data sets and modify contents of displayed **;
                        ** graph, title bar, etc. **;
return;
```

This modification allows nesting of interactive graphs. The Menu Execution Program can create a new graph output, a new IANNO data set, and modify the existing G\_LIST data set. After this program executes, the user is presented with new interactive graph. In contrast, without this modification, the Menu Execution Program can produce only “dumb” (noninteractive) graphs. We have purposely not implemented these changes into our program. Note that linking CLOSE\_DS (close data sets) and OPEN\_DS (open data set) blocks results in an overhead and will slow down the performance. The modification should be made only if nesting of interactive graphs is needed.

## 7. An Example Calculation of XVAL And YVAL Coordinates

The variables XVAL and YVAL stored in the IANNO data set need to be calculated from the original graph data. Recall that (XVAL, YVAL) specify the location of the interactive point as the percentage of the graph output display area. To calculate these coordinates, you will need to identify several GOPTIONS, AXIS, and GDEVICE parameters. In this example, we specified the following:

```
GOPTIONS:
  VORIGIN=0 IN
  HORIGIN=0 IN
  VSIZE=0 IN
  HSIZE=0 IN
  GUNIT=PCT
```

We used the AXIS statements to specify the display parameters for the graph:

```
AXIS1  LENGTH= lenx PCT
        ORIGIN=(orix PCT, oriy PCT)
        OFFSET=(offx1, offx2)
        ORDER= xmin TO xmax;
AXIS2  LENGTH= leny PCT
        OFFSET=(offy1, offy2)
        ORDER= ymin TO ymax;
```

With the above naming convention, the formula for XVAL and YVAL in terms of data coordinates X and Y is as follows:

$$XVAL = orix + offx1 + (lenx - offx1 - offx2) \left( \frac{X - x \text{ min}}{x \text{ max} - x \text{ min}} \right)$$

$$YVAL = oriy + offy1 + (leny - offy1 - offy2) \left( \frac{Y - y \text{ min}}{y \text{ max} - y \text{ min}} \right)$$

## 8. References

*FRAME Application Development Concepts, Version 6, First Edition*, Cary, NC: SAS Institute Inc., 1995.

*SAS Screen Control Language: Reference, Version 6, Second Edition*, Cary, NC: SAS Institute Inc., 1994.

*SAS/AF Software: FRAME Entry Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc., 1993.

## 9. Appendix

```
/*-----
Program:      igrph.scl

Author:       Robert Peszek

DISCLAIMER:  THE SCL PROGRAM AND THE FRAME ENTRY ARE PROVIDED "AS IS"
              WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED,
              INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF
              MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
              RECIPIENTS ACKNOWLEDGE AND AGREE THAT NEITHER AUTHORS
              NOR SAS INSTITUTE SHALL BE LIABLE FOR ANY DAMAGES
              WHATSOEVER ARISING OUT OF THEIR USE OF THIS MATERIAL.
              IN ADDITION, SAS INSTITUTE AND AUTHORS WILL PROVIDE NO
              SUPPORT FOR THE MATERIALS CONTAINED HEREIN.

Description:  SCL source code for igrph.frame
              interactive graph software

input:       dataset work.g_list,
              sas datasets specified by ianno  variable in work.g_list
              gseg entries specified by gout_nm variable in work.g_list

window blocks: prev_pb  : push button
                  clicked event causes displaying next graph specified
                  in work.g_list

                  next_pb : push button
                  clicked event causes displaying previous graph
                  specified in work.g_list

                  show_pb : push button and a char variable
                  hides and displays interactive points

                  graph   : SAS/GRAPH output object
                  displays gplot output specified by gout_nm variable
                  in work.g_list

                  graphic : GRAPHICS object
                  linked to dataset specified by IANNO variable in
                  work.g_list

                  con_box : container box
                  not referenced in SCL, allows graph and graphic to be
                  siblings

non-window variables:
                  named as in G_LIST and IANNO datasets,
                  p_selec, progexe, graph_no, y_var, x_var, clicked

non-window blocks:
                  open_ds : open datasets
                  close_ds : close datasets
                  ini_disp : initialize display settings
                  selCt_pt : executes on select point (on the graph) event
                  mk_popup : executes on popup event
                  submit   : submits menu execution program
                  disp11  : displays _msg_ and gets info about selected point
-----*/

*-----+
| macro initpop used to initialize interactive annotation |
|   variables exe1-exe9 and                               |
|   name_e1- name_e9 used in pop-up menu                 |
+-----+;

%macro INITPOP;
  %do i=1 %to 9;
    exe&i $40 name_e&i $40
  %end;

```

```

%end;
%mend INITPOP;

init:
*-----;
* initialize variables ;
*-----;
length p_selec 8 clicked 8 desc_st $ 40 pnt_id $40
      progexe $40
      gout_nm $40 ianno $40 g_label $40
      %initpop ;
*-----;
* initialize arrays for popup menu of sas programs for selected point ;
*-----;
      array exeprog{9} exe1-exe9;
      array popname{9} name_e1-name_e9;
*-----;
* initialize graph_no so the display starts from 1-st ;
* graph from g_list dataset ;
*-----;
      graph_no=1;
*-----;
* link block opening datasets ;
*-----;
      link open_ds;
*-----;
* link block initializing the display settings ;
*-----;
      link ini_disp;
return;

main:
return;

term:
*-----;
* link close datasets block ;
*-----;
      link close_ds;
return;

open_ds:
*-----;
* this block opens datasets ;
*-----;

*-----;
* open g_list ;
*-----;
      dsid1=open('WORK.G_LIST','i');
      if dsid1 = 0 then _msg_='Error opening WORK.G_LIST';
      call set(dsid1);

*-----;
* load g_list observation no. graph_no ;
*-----;
      rc=fetchobs(dsid1,graph_no);

*-----;
* hide up button if first graph from g_list, unhide otherwise ;
*-----;
      call notify('prev_pb', '_is hidden_', rc);
      if graph_no = 1 and rc = 0 then call notify('prev_pb', '_hide_');
      if graph_no > 1 and rc = 1 then call notify('prev_pb', '_unhide_');
*-----;
* hide down button if last graph from g_list, unhide otherwise;
*-----;
      graph_mx= attrn(dsid1, 'nobs');
      call notify('next_pb', '_is hidden_', rc);
      if graph_no = graph_mx and rc = 0 then
          call notify('next_pb', '_hide_');
      if graph_no < graph_mx and rc = 1 then
          call notify('next_pb', '_unhide_');
*-----;
* specify IANNO and graph output name ;

```

```

*   for GRAPHIC and GRAPH objects          ;
*-----;
    call notify('graphic','_set_dsname_',ianno);
    call notify('graph','_set_graph_',gout_nm);

*-----;
*   change window title bar  ;
*-----;
    if g_label ne ' ' then call wname('Interactive Graph: '||g_label);

*-----;
*   open dataset specified in IANNO ;
*-----;
    dsid2=open(IANNO,'i');
    if dsid2 = 0 then _msg_='Error opening'||IANNO||' specified as IANNO';
    call set(dsid2);
return;

ini_disp:
*-----;
*   this block initializes display settings ;
*-----;
    call notify('graphic', '_set_color_', 'axis', 'white');
    call notify ('graphic', '_set_max_', 'dep_axis', 100);
    call notify ('graphic', '_set_min_', 'dep_axis', 0);
    call notify ('graphic', '_set_max_', 'indep_axis', 100);
    call notify ('graphic', '_set_min_', 'indep_axis', 0);
*-----;
*   link show_pb label with show_pb = ' ' ;
*-----;
    link show_pb;
*-----;
*   modification for SAS 6.12 only ;
*-----;
    call notify('graphic', '_set_color_', 'backplane', 'transparent');
return;

close_ds:
*-----;
*   this block closes datasets ;
*-----;
    rc=close(dsid1);
    rc=close(dsid2);
return;

show_pb:
*-----;
*   block shows-hides interactive points ;
*   clicked push button event executes this block with ;
*   show_pb='Show' or show_pb='Hide' SCL links with show_pb= ' ' ;
**-----;
    if show_pb = 'Show' then
        do;
            call notify ('graphic', '_set_marker_', 13, 1);
            call notify ('graphic', '_set_color_', 'marker', 'green', 1);
            call notify('show_pb', '_set_label_', 'Hide');
        end;
    else if show_pb in ('Hide', ' ') then
        do;
            call notify ('graphic', '_set_marker_', 11, 1);
            call notify ('graphic', '_set_color_', 'marker', 'black', 1);
            call notify('show_pb', '_set_label_', 'Show');
        end;
return;

graphic:
*-----;
*   runs GRAPHIC object events ;
*-----;
    if _status_ = 'P' then link mk_popup;
    else
        if _status_ = ' ' then link selct_pt;
return;

```

```

selct_pt:
*-----;
*   block executes select event on GRAPHIC object   ;
*   get coordinates of selected point               ;
*-----;
    call notify('graphic', '_get_value_', depinfo);
    clicked=listlen(depinfo);
    if clicked > 0 then do;
        lclickx = winfo('xpixel');
        lclicky = winfo('ypixel');
        info= getniteml(depinfo, 'depvalue');
        y_var= getitemn(info, 1);
        call notify('graphic', '_get_value_', indinfo);
        x_var= getnitemn(indinfo, 'indvalue');
*-----;
*   link block displaying info on clicked point as _msg_ ;
*-----;
        link displ1;
    end;
    else _msg_='Try clicking on a point again';
return;

submit:
*-----;
*   submit Menu Exec Prgm, pass info about selected point ;
*-----;
    submit continue;
    %let  pnt_id = %str(&pnt_id);      * clicked point = PNT_ID ;
    %let  desc_st = %str(&desc_st);   * short description   ;
    %let  graph_no = &graph_no;      * graph number       ;
    %let  menu_no = &p_selc;         * menu item no      ;
    %include &progexe;
    endsubmit;
return;

displ1:
*-----;
*   this block displays info on clicked point as _MSG_ ;
*-----;
    rc=where(dsid2,'xval=|| x_var||' and yval=|| y_var);
    rc=fetch(dsid2);
    if rc ne 0 then _msg_ = 'Error: see displ1 block';
    else do;
        _msg_=desc_st; refresh;
    end;
return;

prev_pb:
*-----;
*   block executed by Push Button clicked event   ;
*   moves up on the graph list                     ;
*-----;
    link close_ds;
    graph_no = graph_no -1;
    link open_ds;
return;

next_pb:
*-----;
*   block executed by Push Button clicked event   ;
*   -moves down on the graph list                 ;
*-----;
    link close_ds;
    graph_no = graph_no +1;
    link open_ds;
return;

mk_popup:
*-----;
*   this block executes on GRAPHIC object popup event ;
*-----;
*-----;
*   check if an interactive point was selected   ;
*-----;
    if clicked le 0 then

```

```

do;
  _msg_ = 'Select an interactive point first';
  return;
end;
*-----;
*   check if right-click was on the selected point   ;
*-----;
  rclickx= winfo('xpixel');
  rclicky = winfo('ypixel');
  if abs(rclickx - lclickx) > 10 or abs(rclicky - lclicky) > 10
  then do;
    _msg_ = 'Right-click on the selected point';
    return;
  end;
*-----;
*   refresh message line with info about the selected point   ;
*-----;
  _msg_=desc_st; refresh;

*-----;
*   get the number of nonmissing popup names from the dataset ;
*-----;
  n_exe =0;
  do i=1 to dim(popname);
  if popname{i} ne ' ' then n_exe= i;
  end;
  if n_exe ne 0 then do;

*-----;
*   initialize choice list   ;
*-----;
    choice=makelist(n_exe);
*-----;
*   place popname array on the list   ;
*-----;
    do i=1 to n_exe;
      rc= setitemc(choice, popname{i}, i);
    end;
    * set selected item number to p_selec **;
    p_selec= popmenu(choice);
    if p_selec ne 0 then do;
      *set progexe as selected sas program **;
      progexe = exeprog{p_selec};
      link submit;
    end;
  end;
end;
else _msg_ = 'Pop-up menu not available for this point';
return;

```

Questions and comments should be directed to:

Iza Peszek, PhD  
Merck & Co., Inc.  
P. O. Box 2000, RY33-404  
Rahway, NJ 07065-0900  
E-mail: peszeks@erols.com

Modified code is not supported by the authors or SAS Institute.

SAS, SASAF, SAS/GRAPH and SAS/INSIGHT are registered 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.

Reprinted with permission from *Observations*®.

© 1997 SAS Institute Inc.