

Quick & Easy AF Applications Using SAS/ASSIST® Software, SAS® Data Views, Block Menus and HTML

Jeanne Spicer
The Pennsylvania State University

Abstract

The paper describes techniques used to develop a user-friendly SAS application for accessing a large archive of data files at a university research center. The application was developed on a UNIX network, but can be ported to graphical or non-graphical platforms with little modification. No prior SAS/AF® experience is required.

Introduction

Our university research center wanted to develop custom applications to make data extraction easier for the faculty & students, but were unable to get too far with the plan due to many factors including:

- the large number of unrelated, low-use data sets on the system
- the variety of programming languages used to process the data
- transient nature of programming staff who are unable to invest time in learning to develop custom applications
- varied levels of end-user experience both initial & anticipated

A simple general purpose application created at our site -- known as 'PRI PopUps' -- was designed to meet the following requirements:

- Reduction of user frustration
- Ease of creation & maintenance for programming staff
- Flexibility for the user
- Storage of raw data in compressed ASCII format.

With version 6.09 of the SAS System for UNIX we really started to promote the use of SAS/ASSIST software. Since ASSIST exists on all platforms, the small outlay in time to learn to use it is not wasted. Users have 'point & click' access to unfamiliar data sets without needing highly trained staff to write custom GUI applications. But the data still need to be read into a SAS data set. If we could make the data ready to use with SAS/ASSIST, most of our users' basic needs would be met with a minimum strain on the programming staff.

Data collections are stored on disk on our UNIX network in compressed ASCII format. Prior to the development of the application, users had to submit a SAS DATA step program to input the raw data and create a SAS DATA set. Problems resulted as code was passed around and modified by inexperienced users. Storing SAS data files was impossible due to space considerations and the need to allow non-SAS users access to the data.

SAS DATA step views of the raw data were the solution to the

problem. Views are often referred to as 'virtual' data sets. Raw data is read by a DATA step program, but only the descriptor portion is stored in the SAS file, the actual data values are not copied from the raw file until the view is used. The view can be used directly in ASSIST, by any PROC or by a DATA step SET statement. The raw data is 'SASed on the Fly' as it were. The advantages are that the user does not have to run a DATA step program and that there is a considerable reduction in storage requirements. Consider one of our raw data files stored as a compressed ASCII file in 23,472 bytes. When the data was read into SAS a data set using the COMPRESS=YES option it required 3,612,672 bytes to store the 1113 observations and 833 variables. Storing a SAS DATA step view definition of the data required 405,710 bytes. The total, including the ASCII raw data file, is still only 429,182 bytes. That's about 78% less disk space than the data set. The trade-off is that the access time is longer. Creating an extract of 7 variables from the data set takes about 5 seconds, while it takes about 480 seconds to pull the data through the view to create the same extract. Since this is a one time run, we opted for the savings in disk space.

Our FORTRAN programmers are happy because the raw data is still there, our system administrator is happy because storage requirements are reduced and, most importantly, our SAS users have direct access.

That forms the basis of the 'PopUps', now all that was needed was a little 'window dressing' :-)

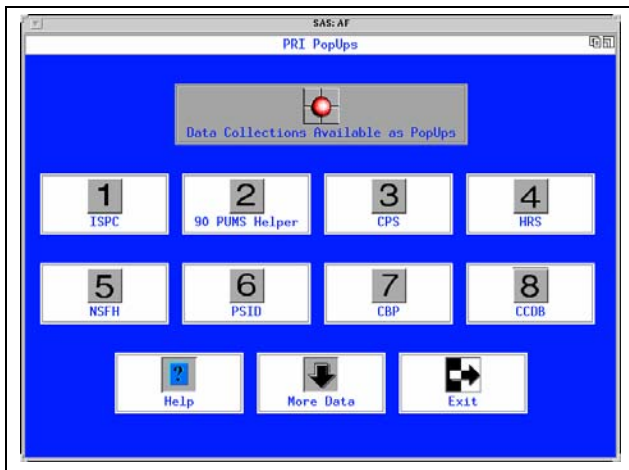
An index to the data collections on our network had been posted on our web site homepage in Hypertext Markup Language (HTML). Why not take advantage of this by providing a button to link to these documents within SAS? Additional documentation about the contents of the SAS DATA step views and the use of the 'Popups' could be added and accessed from inside or outside of the application.

A simple, standardized front-end that uses BLOCK menus and CHOICE groups collects all these pieces together to give the users access to the SAS data files, appropriate documentation and SAS/ASSIST. The block menus are created by an SCL program that is easy to understand and modify. It's right off the shelf on pages 241-244 of the SCL reference manual. [see references]

Using the PopUps

To use the PopUps, users copy 2 statements into their autoexec.sas files; a LIBNAME statement, to make the appropriate directory available and an AF command to fire up the application.

When the user invokes a SAS session, the 'PRI PopUps Main Menu' window appears on top of the usual SAS Display Manager windows.



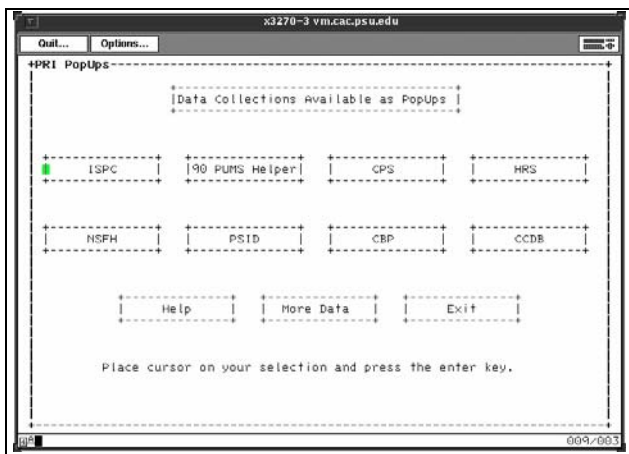
The main menu displays a button for each data collection.

The 'Help' button brings up a screen with some instructions and the full title of each data collection.

The 'More' button [optional] brings up other pages of the main menu if needed. Only 12 buttons per page are allowed in a block.

The 'Exit' button returns the user to display manager.

The application works without much modification in a non-



graphical environment as well. The display below shows how the main menu looks on an IBM 370 running the SAS System for CMS. When one of the data collection buttons is pressed all LIBNAME & FILENAME statements are issued and a sub-menu appears. The sub-menus for each data collection have the same general format, see Figure 3.

The 'Documentation' button opens a Netscape window pointing to a local file containing a description of the data file with links to the output of PROC CONTENTS and descriptive statistics from PROC MEANS or FREQ and FMTLIB output, if any. There is a link to a 'How to Extract Data' document providing step-by-step instructions on creating an extract with SAS/ASSIST. You could To modify a DATA step program that creates a SAS data set from a raw file into one that creates a SAS data view simply add the

alternatively construct a document with CBT frames, a text editor or word processing software.

The 'Copy libnames' button copies the code for the LIBNAME and FILENAME statements into the user's home directory. This code can be submitted in any display manager session, inserted into batch code or attached to a program generated by SAS/ASSIST.

The 'SAS/ASSIST' button invokes SAS/ASSIST software. See

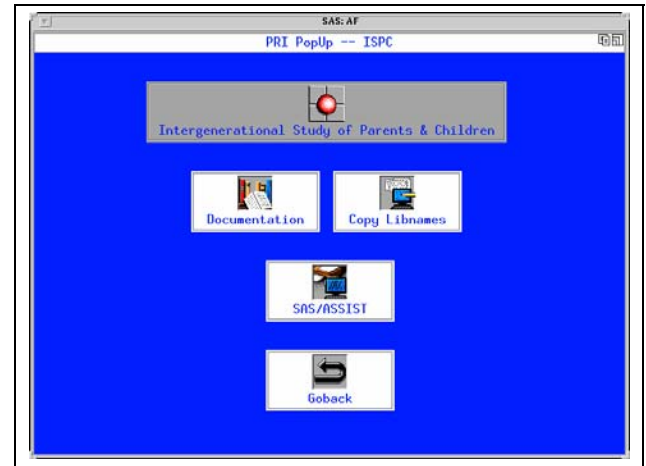


Figure 5 for a look at the user's work area.

The 'Goback' button returns the user to the main menu.

How to Create 'PopUps'

Code for all the modules in this application is provided as an appendix to this paper. The examples are written for the UNIX environment. Code is also available via ftp (see notes) for UNIX, WINDOWS and CMS.

Use the code as a template for the construction of your own application. A copy of the SAS/AF Software manual will be useful if you have never used the AF BUILD screens. [see references]

Step 1 Directories

Create a directory for the application catalog (this will be referred to as the 'PopUp directory'). Copy the sample code to this directory. Create a sub-directory in this directory for each logical collection of data (these will be referred to as 'data collection sub-directories'). A data collection sub-directory will contain the DATA step views and any other related code like format libraries, batch code, etc. Existing raw data files and documentation can stay where they are.

Step 2 DATA step view definition

The DATA step code to create a SAS data view requires very little modification to a standard DATA step program. The data step program can include any valid DATA step statements: recodes, FORMAT assignments, MERGE statements, etc.

VIEW= option to the DATA statement.

```
DATA libref.filename / view=libref.filename;
```

When accessing the view, both the FILEREF pointing to the raw data and the LIBREF for the view are required as the view does not actually contain any data values. An excellent article on views by John Boling appears in 3rd Quarter 1993 issue of *Observations*. [see references]

Step 3 Storing LIBREFS & FILEREFs

Store the FILENAME statement required to access the raw data and the LIBNAME statements to access the DATA step views and format libraries as a SAS program in the data collection sub-directory. When the data collection is selected from the main menu, this program will be submitted to make the references active. The user can also attach these statements to their own jobs by pressing the 'Copy libnames' button in the data collection sub-menu. See sample program #1 -- REFS.sas

Step 4 HTML Documentation

Create an HTML document describing the data and file it in the data collection sub-directory. A good introduction to HTML can be obtained from NCSA [see references]. This document could contain links to output from PROC CONTENTS and other descriptive statistical procedures which can be stored as separate files in the data collection sub-directory. Until there is a built-in facility for generating HTML output, you can use a simple block of title statements to generate HTML tags for the output of a SAS procedure. See sample program #2 -- HTMLTAG.sas for an example of a program that produces PROC CONTENTS output ready for an HTML browser.

Step 5 Building the Main Menu with an SCL Entry

Start a display manager session and issue a LIBNAME pointing to the 'PopUp' directory.

```
libname popup 'xxxxx';
```

Create the new AF application catalog called 'startup' in the 'popup' library by issuing the command:

```
build popup.startup
```

In the BUILD: DIRECTORY window select

```
File --> New
```

Supply the entry name MAINMENU and choose SCL for the entry type.

In the BUILD: SOURCE window select

```
Open --> Read File
```

Access sample program #3 -- MAINMENU.scl. You will use this program as a template. The Main section of the program includes the BLOCK function which defines a block menu containing 3 rows of 4 buttons each and an associated CHOICE group SELECT statement which associates actions to each of the 12 buttons. Although you do not have to use all 12 blocks, you must account for each one in the BLOCK function. You will

Step 8 Invoking the application

notice that the labels for unused buttons are indicated with a null string and that the icon definition for an unused button is '0'. Edit the program to execute an AF command calling a sub-menu SCL entry that you will build in step 9 for your data collection. Compile the code by selecting

```
Locals --> Compile
```

When the program compiles cleanly, preview it by selecting:

```
Locals --> Testaf
```

When you are satisfied, select

```
File --> End
```

This will save the 'PopUp' main menu definition and return you to the BUILD:DIRECTORY window.

Step 6 Using PROC PMENU

Create some standard buttons to be used in several windows of the application with PROC PMENU.

From the display manager Program Editor window, select

```
File --> Read File
```

and include sample program #4 -- PMENU.sas. The program creates two different sets of buttons. One contains two buttons: Continue & Cancel; the other contains one button: Return. Submit the job and move to the BUILD: DIRECTORY window and you will see two PMENU entries in the catalog. The buttons cannot be viewed until they are associated with a display window. The first window to use a PMENU entry is the main menu 'Help' window.

Step 7 Building a Help Entry

Create a 'Help' screen to provide the user with an overview of the application.

In the BUILD: DIRECTORY window select

```
File --> New
```

Supply the entry name MAINMENU and select the entry type HELP. The BUILD:DISPLAY window will appear. Type the help information on the screen or include a text file. Further define the appearance of the window by selecting

```
Locals--> Set General Attributes
```

from the menu bar. In the BUILD:GATTR window, supply a name to appear as the banner for the 'Help' window and in the *Command Menu* field type: HELPBTN. This will access the 'Return' button defined by PROC PMENU created in step 6. For the *Window type* select DIALOG. A HELP entry has no source code to compile, but you can preview it by selecting

```
Locals --> Testaf
```

To invoke the application you will need to issue the AF command:

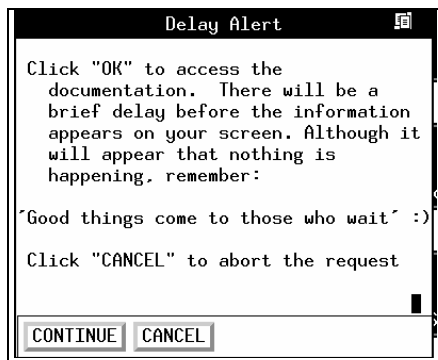
```
af c=popup.startup.mainmenu.scl
```

You may want to edit an autoexec file that will automatically invoke the application each time you start a SAS session. See sample program #5 -- AUTOEXEC.SAS.

Step 9 Building the Sub-Menu with an SCL Entry

Return to the BUILD: DIRECTORY window follow the instructions in step 5 to create another SCL entry. This time use the code supplied in sample program #6 -- DATAMENU.scl as a template. Edit the program to reflect the location of the data collection sub-directory, compile & test it.

Step 10 Building the Documentation button with a PROGRAM



Entry

Documentation in our application is viewed through the Netscape HTML browser. A small program notifying the user of a delay while Netscape fires up is provided by this entry:

In the BUILD: DIRECTORY window create a new entry by selecting

```
File --> New
```

Supply the entry name xxxxHTML (where xxxx is a nickname for your data files) and choose PROGRAM for the entry type.

PROGRAM entries have two parts; a DISPLAY window where you can enter text that you want to appear when the entry is executed and a SOURCE window which contains SCL code for the actions to be performed by the entry.

In the BUILD: DISPLAY window, type in a message to alert the user of a potential delay and give them the option to cancel the request. Bring up the BUILD:GATTR window by selecting Locals --> Set general attributes. Supply a name to appear as the banner for the help window and in the *Command Menu* field type: CONCAN. This will access the 'Continue' & 'Cancel' buttons

defined by PROC PMENU created in step 6. For the *Window type* select DIALOG.

In the BUILD: SOURCE window, from the pull-down menu bar, select: Open --> Read File and access the code supplied in sample program #7 -- DATAHTML.program. Edit the program to include the appropriate libname & filename statements for your data.

Step 11 The Copy Librefs button

The user can obtain a copy of the libname & filename statements that are required for the data collection by pressing this button. Edit program DATAMENU.scl to execute a host command to copy a file from the data collection sub-directory to the user's directory when choice 2 is pressed.

Step 12 Adding additional applications

Once you have the 6 catalog entries, you can really start to crank out the applications. Prepare your views and documentation and in the BUILD: DIRECTORY select:

```
File --> Open --> Copy an object
```

And copy and edit the DATAMENU SCL entry and the DATAHTML PROGRAM entry to access the new data.

Select MAINMENU.SCL and edit the choice block to include a button for the new application and the appropriate AF command.

Conclusion

As your familiarity with SAS GUI applications increases, you will certainly want to build on the application. Maybe you would rather select data from a list than a block menu or maybe a custom built FRAME application would better suit your needs.

For us, this application provided users with a helpful interface to our data collection quickly. And it was easier to persuade management to grant time & money on future development with a working prototype to demonstrate.

References

Boling, John C. (1993), "SAS® Data Views: A Virtual View of Data" *Observations™*, 2(4), 35-41.

Fecht, Marje (1995) "Getting Started with Application Development Using SAS/AF® Software".

National Center for Supercomputing Applications "Beginner's Guide to HTML", NCSA, <http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html>

SAS Institute Inc., *SAS/AF® Software: Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc., 1989, 245p

SAS Institute Inc., *SAS® Screen Control Language: Reference, Version 6, Second Edition*, Cary, NC: SAS Institute Inc., 1994, 648pp.

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

For more information

Contact the author:

Jeanne Spicer
 The Pennsylvania State University
 813 Oswald Tower
 University Park, PA 16802
 spicer@pop.psu.edu

Code referred to in this paper is available by anonymous FTP:

```
ftp ftp.psu.edu
<user> anonymous
<passwd> your email address
cd incoming/sugi21
mget *.*
quit
```

Appendix

```
/*-----*/
Sample program #1
Issue librefs & filerefs          REFS.sas
/*-----*/
Libname statement associates a libref
with the directory location of SAS view
Views require fileref to raw data
Note: The raw data is uncompressed and
piped to SAS when the view is accessed.
This program will be executed when the
ISPC sub-menu is requested and can be
copied by the user by pressing the 'Copy
libnames' button in the sub-menu.
/*-----*/
libname ispcview '/home/pop/ispc';
filename ispcm62
pipe'zcat /home/share/da9902.w6277.Z';
run;

/*-----*/
Sample Program # 2
Insert HTML tags in SAS output    HTMLTAG.SAS
/*-----*/
Options define a longer page size, and remove
page numbers & dates from headings.
The <PRE> tag indicates that the format of the
should be honored by the browser. Should you
want to attach something after the output
you will need to insert a </PRE> tag before
the additional text.
<TITLE> defines the banner on the window
<H2>&<H4> define heading & sub-heading in bold
and larger type.
/*-----*/
options ls=72 ps=500 nonumber nodate;
title1 '<PRE> <TITLE>Window Banner</TITLE>';
title2 '<H2> Heading </H2>';
title3 '<H4> Sub-heading </H4>';
proc contents data=ispcview.mother62 ;
run;

/*-----*/
Sample program # 3
PopUp Main Menu                   MAINMENU.SCL
/*-----*/
/*-----*/
The init section issues a libname
for a temporary directory for the
user to write to during the session
It is optional.
/*-----*/
```

```
init:
  if (libname('sastmp','/sastmp'))
    then _msg_=sysmsg();
return;
/*-----*/
| The main section defines a button
| for each data collection sub-menu,
| a help screen and an exit.
/*-----*/
main:
choice=1;          /* Initialize choice */
/* process menu choices until Exit(button 12) */
/* is pressed                                           */
do until(choice=0 or choice=12);
  choice=block('PRI PopUps', /* Window Banner */
'Data Collections Available as PopUps', /*Title*/
  14, /* Color scheme */
/* Labels */
'ISPC', '1990 PUMS', ' ', ' ', /* 1st Row */
'NSFH', ' ', ' ', ' ', /* 2nd Row */
' ', ' ', 'Help', 'Exit', /* 3rd Row */
/* Icons */
141,142,143,144, /* 1st Row */
145,146,147,148, /* 2nd Row */
0,0,270,111); /* 3rd Row */

  select(choice);
  /* Add WHEN clauses for each data collection */
  /* Display ISPC sub-menu */
  when(1)
    call display ('ispcmenu.scl');
  /* Display Custom Help Screen */
  when(11)
    call display ('mainmenu.help');

  otherwise;
  end;
end;
return;
/*-----*/
| The Term section is optional
/*-----*/
term:
return;

/*-----*/
Sample program # 4
Make buttons for Display windows  PMENU.SAS
/*-----*/
/*-----*/
The CONTINUE button executes a command in the
DATAHTML.PROGRAM invoking Netscape.
The CANCEL button issues an 'end' command to
cancel the request for documentation.
/*-----*/
proc pmenu c=popup.startup desc 'PMENU.SAS';
menu concan;
item 'CONTINUE' selection = i;
item 'CANCEL' selection=e;
  selection i '_OK';
  selection e 'end';
run;
/*-----*/
| The Return button issues an 'end' command to
| exit the Main Menu help screen.
/*-----*/
menu helpbtn;
item 'Return' selection = r;
  selection r 'end';
run;
QUIT;

/*-----*/
Sample program # 5
Autoexec file statements          AUTOEXEC.sas
/*-----*/
Assign libref for application directory
and issue af command to open the
application when the Display Manager
```

```

| session is invoked.The af command is a
| display manager command & must be pre-
| ceded by dm when issued outside of
| a display manager session.
|-----*/

```

```

/*-----*/
| Sample Program #6
| Data Collection Menu DATAMENU.SCL
|-----*/

```

```

| The init section calls the code
| which issues librefs for the SAS
| data views and the associated
| filerefs for the raw data.
|-----*/

```

```

init:
rc=filename
('REFS','/home/pop/ispc/ispcREFS.sas');
if (rc ne 0) then _msg_=sysmsg();
rc=preview ('include close','REFS');
submit immediate;
endsubmit;
rc=preview ('clear');
return;

```

```

/*-----*/
| The main section defines buttons
| for copying those reference to
| the users own directory,accessing
| the documentation and invoking
| SAS/ASSIST.
|-----*/

```

```

main:
choice=1;
do until(choice=0 or choice=9);
choice=block('PRI PopUp -- ISPC',
'Intergenerational Study of Parents & Children',
14,
'Documentation', 'Copy Libnames',' ',' ',
'SAS/ASSIST',' ',' ',' ',' ',
' ',' ',' ',' ', 'Goback',
7,5,0,0,
404,0,0,0,
0,0,0,116);
select(choice);
when(1) /* Access pgm to read documentation */
call display
('popups.startup.isphtml.program');

```

```

libname popup '/home/pop';
dm 'af c=popup.startup.mainmenu.scl';

```

```

when(2) /* Host command to copy refs */
rc=system
('cp /home/pop/ispc/ispcREFS.sas ~/.');
when(5) /* Invoke SAS/ASSIST software */
call execcmd ('assist');
otherwise;
end;
end;
return;

```

```

/*-----*/
| Sample Program #7
| Access HTML documentation DATAHTML.PROGRAM
|-----*/
| Execute host command to invoke HTML browser &
| display a message window alerting the user
| of a delay.
|-----*/

```

```

length command $3;
init:
control always;
return;
main:
command=word(1,'u');
call nextcmd();
if command='_OK' then do;
/* Start netScape as a background process */
rc=system
('netscape /home/pop/ispc/ispc.html &');
_status_='H';
if (rc ne 0) then _msg_=sysmsg();
end;
return;
term:
return;

```

SAS: OUTPUT

SAS: LOG

File Edit View Globals Help

```

Physical Name: /sastnp
NOTE: Libref PARCHILD was successfully assigned as follows:
Engine:
Physical Name: /home/data1/axinn/views
NOTE: Libref PRIPOPUP was successfully assigned as follows:
Engine:
Physical Name: /home/data4/spicer
NOTE: AUTOEXEC processing completed.

```

SAS: PROGRAM EDITOR

File Edit View Locals Globals Help

```

00001
00002
00003

```

SAS: ASSIST

SAS/ASSIST


Primary Menu

TUTORIAL	DATA NIGHT	REPORT WRITING	GRAPHICS
DATA ANALYSIS	PLANNING TOOLS	EIS	REMOTE CONNE
RESULTS	SETUP	INDEX	EXIT

Intergenerational Study of Parents and Children, 1962-1985 [Detroit]

File Edit View Go Bookmarks Options Directory Help

Welcome What's New What's Cool Questions Hot Search



Intergenerational Study of Parents and Children, 1962-1985 [Detroit]

- [About the Study](#)
- [SAS data files: Contents & Means](#)
- [How to Access PFI PopUps](#)
- [How to Create an Extract](#)

About the Study

Investigators: Ariand Thornton and Debraah Freedman

Summary: This data collection provides information of family formation and dissolution among young adults. Families who had given birth to their first, second or fourth child in 1961 comprised the group of Detroit-area Caucasian couples who were interviewed and surveyed over the period 1962-1985. The resulting longitudinal study encompasses six waves of data collected from mothers across the entire span of their offspring's childhood. Included are demographic, social, and economic information about the parental family, information about the attitudes, values, and behavior of both the mother and the father, and information about the mother's desires and expectations for her child's education, career