

KEEP IT SIMPLE: APPLIED SAS/AF MENU TECHNIQUES

Vicentee Ferguson, Blue Cross and Blue Shield of Florida

Abstract

By using SAS Application Facility (SAS/AF®) and Screen Control Language (SCL), program developers can make user-friendly and user-proof menuing systems. This paper will demonstrate some basic screen design and SCL techniques, with an emphasis on creating block menu applications. A brief discussion of the need for menu systems which are easy for users to understand and operate will be followed by more detailed documentation of how to use SAS/AF and SCL to create attractive, efficient, and developmentally simple menu applications. This discussion will include an overview of parameter passing and substitution techniques that can make SCL code efficient and flexible.

Introduction

Blue Cross and Blue Shield of Florida (BCBSF) is the state's largest health insurance company, providing managed care, administrative services-only (claims processing), and traditional indemnity type products. Additionally, BCBSF manages the federal Medicare contract within the state of Florida. Managing such a wide variety of products for a large customer base, creates some unique data processing (DP) challenges. One such challenge is to get the appropriate data to business analysts in a form they can use and in a time frame that is meaningful.

We developed a prototype Statistical Information System (SIS) using SAS/AF which provides on-line access to data which was previously available to analysts only in periodic hard-copy reports or by ad-hoc request. The SIS, or similar user-friendly information system, can reduce the need for standardized paper reports and ad-hoc requests at the same time they give users the ability to extract from the system the data they really need, not just what is in a standard report.

The Need For Menuing Systems

A business analyst's tasks include gathering and analyzing data contained in a company's information system (and possibly from external sources), and making recommendations for improvements. Analysts usually do not have time or necessarily the specialized data processing (DP) skills to

obtain, and organize large amounts data. However, out of necessity, some analysts become programmers or perform other DP functions in order to get at the data they need. This may not be the best use of their time and may be outside the ambit of their specialized skills. The analyst/programmer's lack of DP experience may also lead to data integrity problems. Menuing systems can provide analysts with easy access to the data, thus allowing them more time to use their analytical skills, while leaving DP functions to DP professionals.

To be effective, a basic information system should possess two critical characteristics: 1) allow non-programmers uncomplicated access to data, and 2) have a user-friendly interface. A user-friendly interface is an intuitive menu system. (More complex systems should provide routine analysis to help the analyst focus on problem areas and allow him/her to "drill down" on the problem area to find its root cause.) A truly intuitive menu system is "self documenting" in the sense that a user should be able to figure out how it works (at least the major functions) by just "playing around" with it. Written documentation is for the more complex tasks.

SAS/AF has proven to be an excellent choice for developing menu systems because it makes user-friendly applications fairly easy to develop, and because the analytic capabilities of base SAS® are diverse and powerful. Another advantage is portability. Base SAS and SAS/AF software can be used in both the PC and mainframe environments. Further, SAS/AF provides the developer with a number of menuing options (e.g., push button, choice groups, selection lists, block menus, etc.). We will focus on the block menu.

Getting Started

The code required to create and use block menus is stored in the source window of a SAS/AF program entry. Thus, the first steps required of you are to allocate a library in which to store the program entry, and then create a catalog and program entry using the proc build statement (see SAS® *Screen Control Language - Usage Version 6* for details) The basic form of the statements to accomplish these tasks are:

```
LIBNAME libref 'data-library';
```

```
PROC BUILD C=libref.catalog.entryname.program;
```

When the BUILD statement executes, a display window will appear. Enter "source" at the command line to bring up the source window. This is where you will enter code related to block menus.

Block Menu

A block menu is a user-friendly display with block-like icons that represent different tasks a user can perform. The user simply places the cursor (using the TAB key or mouse) on the appropriate block and presses enter to select the desired task. One advantage of this kind of menu is that it is easy to use and understand. Additionally, using block menus requires no typed input, thus eliminating the possibility of user input errors or the need for the developer to control for acceptable input (e.g., McMillan vs. MCMillan vs. MCMILLAN vs. McMilan). However, the developer must carefully think through and design each display screen to ensure that all necessary choices are shown on the screen.

Block menus are created using the BLOCK function. Each block menu can display up to three rows of four 4 blocks each. The basic form of the BLOCK function is:

```
choice=BLOCK(window-name,title,color,text1,...,text12);
```

Choice is a numeric variable containing the number of the block a user selects. *Window-name* is text that appears in the upper left corner of the display screen, while the *title* is text that is displayed in the screen's title box. *Color* is a number (0-31) representing the desired color scheme of selection blocks. Finally, *text1*,...,*text12* indicates the text that appears in each selection block.

SAS/AF version 6.07 has more than a hundred pictorial icons that can be used with the BLOCK function instead of the default block-like icons. A typical BLOCK function with pictorial icons might look like this:

```
choice=block('Science Menu','SCIENCE AND
TECHNOLOGY',20,'LABWORK',
'DESIGN TESTING','DBMS','UTILITY',
'ANOVA','SQL','SYMBOL','ANALYSIS',
'MAPS','COUNTS','SETUP','EXIT',50,55,
120,123,134,34,29,105,131,126,109,111);
```

The numbers 50 and 55 identify the pictorial icons which represent the lab work and design testing tasks. The text in *text1*,...,*text12* appears below the associated pictorial icon. In the example above, "DESIGN TESTING" would appear below pictorial icon number 55.

BLOCK functions really come to life when they are combined with appropriate SCL in a program entry. Figure 1.1 shows a block menu display that allows a user to select which marketing segment he/she is interested in. Figure 1.2 is the SCL coding that produced Figure 1.1.

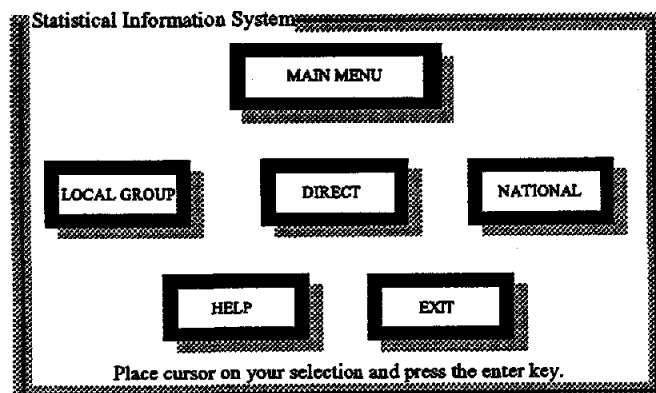


Figure 1.1

```
INIT:
return;
MAIN:
CHKVAR=1;
do while(CHKVAR ne 0);
CHKVAR=block('Statistical Information System',
'MAIN MENU',7,'LOCAL GROUP',
'DIRECT','NATIONAL','',
'', 'HELP', '', 'EXIT', '');

select(CHKVAR);
when(1) call display('LGMAIN.PROGRAM');
when(2) call display('DIRMAIN.PROGRAM');
when(3) call display('NATMAIN.PROGRAM');
when(9) call display('MAIN.HELP');
when(11) call execcmd('bye');
otherwise;
end;
end;
return;

TERM:
return;
```

Figure 1.2

The default cursor location is in the upper left corner of the display window. By setting the variable CHKVAR equal to 1, the cursor will appear in the Local Group selection block when the menu is displayed. If users work with the Direct market segment more often, it might be appropriate to set CHKVAR equal to 2.

The BLOCK function returns a numeric value based on the user's menu selection. In Figures 1.1 and 1.2 above, if National were selected, the BLOCK function would return a numeric value of 3 which is stored in CHKVAR. The value of CHKVAR is 0 if an END or CANCEL statement is submitted. The SELECT statement queries CHKVAR and executes the appropriate SCL code based on CHKVAR's value. In the example above, choosing National causes the natmain.program entry to be executed.

A program entry called and executed by the CALL DISPLAY statement might perform any number of tasks. For example, it might print a contents or observations from a data set, display a report or help screen, or even display another block menu. Using a BLOCK function and SELECT statement within a DO-WHILE loop causes the block menu to redisplay after the code linked to the user's menu choice has been executed. Otherwise, the block menu would close after performing the task associated with the user's menu selection, and execute any SCL code that follows the block function. Whenever you have a program entry which does not have defined screen variables (as with the block menu display) the MAIN: section executes only once. Without the DO WHILE and SELECT statements, the block menu would close after performing the selected task, execute SCL code remaining in the MAIN: and TERM: sections, then the entire program entry would close. The DO WHILE and SELECT statements keeps executing the block menu until the user submits an END or CANCEL statement.

Figures 2.1 and 2.2 show several useful techniques. Occasionally you may need to pass parameters derived in one program entry to another. The SCL code in Figure 2.1 accomplishes this by creating the character variable MARKSEG. MARKSEG is assigned a value based on the selection block a user chooses. The value of MARKSEG is passed to another program entry by including MARKSEG in the CALL DISPLAY statement. A corresponding statement is needed in the program being called to receive the passed parameters. The ENTRY statement in the INIT: section of Figure 2.2 receives the passed parameter.

Parameter passing and substitution are very effective techniques for reducing the amount of SAS and SCL coding

```
INIT:
length MARKSEG $10;
return;

MAIN:
choice=1;
do while(CHKVAR ne 0);
CHKVAR=block('Statistical Information System',
             'MAIN MENU',7,'LOCAL GROUP',
             'DIRECT','NATIONAL',' ',' ',' ',' ',
             'HELP',' ','EXIT',' ');

select(CHKVAR);
  when (1) do;
    MARKSEG='Local Group';
    call display('LGMAIN.PROGRAM',MARKSEG);
  end;
  when (2) do;
    MARKSEG='Direct';
    call display('DIRMAIN.PROGRAM',MARKSEG);
  end;
  when (3)
    MARKSEG='National Accounts';
    call display('NATMAIN.PROGRAM',MARKSEG);
  end;
  when (9) call display('MAIN.HELP');
  when (11) call execcmd('bye');
  otherwise;
  end;
end;
return;

TERM:
return;
```

Figure 2.1

in a program entry. In figure 2.2, SISD.SYRMON is a small SAS data set that contains the character variable YRMON. The values of YRMON are 9201, 9202, etc. (where 9202 represents February 1992). After the SYRMON data set is opened, the DATALISTC function opens a window that displays all the values of the YRMON variable in the data set and allows the user to select one. The selected YRMON value is stored in the SCL variable DATE.

SAS statements can be submitted from within a SAS/AF program entry by placing them between the SUBMIT and ENDSUBMIT statements. The statements shown in figure 2.2 between the SUBMIT CONTINUE and ENDSUBMIT statements are regular SAS coding. There are several forms of the submit statement. See *SAS® Screen Control Language Reference Version 6* to determine the appropriate

form for your application. Notice the ampersand attached to the variable DATE in the IF statement. When the statement is read by SAS, it will evaluate to - IF YYMM="9202" - (assuming 9202 was the date selected from the DATALISTC). This IF statement selects observations from the SISD.ALLDATA dataset that have YYMM values equal to 9202. Note also that the value of DATE and MARKSEG are substituted into the proc report titles. Substitution makes SCL and SAS code flexible, efficient, and easy to read. Rather than writing a number statements to handle all possible choices of dates, the IF statement in Figure 2.2 can handle any date a user selects.

```
INIT:
entry MARKSEG $10;
YEAR=open('SISD.SYRMON');
call wregion(10,30,15,41,11);
DATE=datalistc(YEAR,'YRMON','Choose a Year/
               month For Analysis');
rc=close(YEAR);
return;

MAIN:
submit continue;
options nodate;
data sisone;
    set sisd.alldata;
    if yymm="&DATE";
proc report ps=20 data=sisone
report=sisp.org.procline.rept;
title1 "&MARKSEG Market Segment";
title2 "End Of The Month Inventory For &DATE";
Footnote 'PF3 - RETURN, PF7 - UP, PF8 - DOWN';
run;
endsubmit;
return;

TERM:
return;
```

Figure 2.2

Passing parameters can be a bit tricky if multiple parameters are involved. Parameters must be listed in the same order and have the same type (e.g., numeric, character) in both the CALL DISPLAY and ENTRY statements. If you experience errors passing parameters, the problem will likely be that a different number of parameters were passed than received, or the order of passed and received parameters is different.

We often use parameter passing techniques to modify a menu's title block based on the selection block the user chose at a previous block menu display screen. The portion of the SCL code required to accomplish this is shown below in figures 3.1 and 3.2

```
INIT:
length MARKSEG $10;
return;

MAIN:
choice=1;
do while(CHKVAR ne 0);
CHKVAR=block('Statistical Information System',
             'MAIN MENU',7,'LOCAL GROUP',
             'DIRECT','NATIONAL',' ',' ',
             ' ','HELP',' ','EXIT',' ');

select(CHKVAR);
when (1) do;
    MARKSEG='LOCAL GROUP';
    call display('LGMAIN.PROGRAM',MARKSEG);
end;
end;
return;
other SCL
```

Figure 3.1

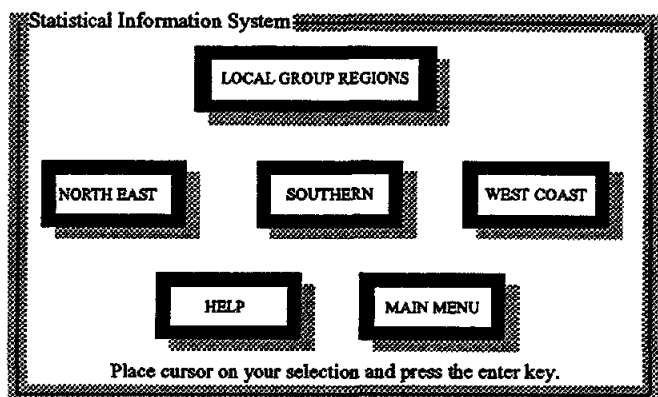
```
INIT:
entry MARKSEG $10;
return;

MAIN:
choice=1;
do while(CHKVAR ne 0);
CHKVAR=block('Statistical Information System'),
            MARKSEG||' REGIONS',7,
            'NORTH EAST','SOUTHERN',
            'WEST COAST',' ',' ',' ',
            ' ','HELP',' ','EXIT',' ');

end;
return;
other SCL
```

Figure 3.2

Here is the result:



Summary

Menu systems are not a panacea for all data processing ills. However, they are most useful in situations where data users are not programmers and DP professionals are relied upon to handle most adhoc requests for data and reports. SAS/AF's BLOCK function provides developers with a simple way of creating user-friendly menus. Using this function with appropriate SCL, you can create powerful and efficient menu systems. The programs you develop will probably be more complex than the examples shown here. But, these examples demonstrate how easy it can be to develop attractive menus SAS/AF block menus.

Acknowledgments

SAS is a registered trademark of SAS Institute Inc.

The author would like to thank Dwight Scott for his assistance in reviewing this paper.

References*

SAS/AF® Software - Usage and Reference, version 6
 SAS® Screen control Language - Reference, version 6
 SAS® Screen control Language - Usage, version 6
 SAS® Technical Report P-199
 SAS® Technical Report P-216

*This list may be incomplete. Contact SAS Institute, Inc. for a complete list of references on SAS/AF, and SCL.

Contact

For more information contact the author at:

8928 Freedom Commerce Parkway
 FC1-4
 Jacksonville, FL 32256