Getting Started with SAS/AF™ and Doing It Right

Howard Levine, Levine Software Systems

Introduction

SAS/AF™ is a very effective tool for making SAS® applications user friendly. The purpose of this paper is to acquaint you with some ideas that will help you get started when you begin using SAS/AF.

The topics that are covered in this paper are ideas to think about when using SAS/AF,SAS/AF procedures, the types of screens in SAS/AF, useful techniques, and examples.

SAS/AF is used to make screens for users that convey information and prompt a user to fill in the blanks. By filling in the blanks and submitting the screen, SAS code is generated based on what the user typed to fill in the blanks. This is valuable for experienced programmers as well as for non-programmers because it is often easier and more reliable to have automatic procedures set up for tasks than to have to write code each time the task has to be performed.

General Ideas

Some principles that apply to programming in general or to SAS programming also apply to SAS/AF. In addition, there are some ideas that apply only to SAS/AF.

With any software, there will be conflicts between User friendliness and ease of coding. There is also a choice between hard coding items and using parameters. Some choices peculiar to SAS/AF are deciding whether to use program or menu screens for a particular purpose and whether to let AF generate SAS code or have a macro do it.

Initially, you may choose simple coding techniques over more sophisticated coding techniques. When you are getting started with SAS/AF, you will have to do things simply until you better understand screen building techniques. However, do not lose sight of the fact that as you get more experience using SAS/AF, you will start to use more sophisticated techniques. For that reason you should be very careful that you do not set coding standards for your systems that you will soon outgrow. Standards are very useful for keeping maintenance easy. Bad standards can be more trouble than they are worth. Your standards should work for you to enhance your creativity and productivity; they should not inhibit your ability to get the job done. Standards should be able to accommodate sophisticated programming techniques as well as "beginner" systems. If your standards embody beginner system techniques, then you will be condemning yourself to a future filled with awkward systems that look like beginners wrote them regardless of the abilities of the programmers who write your systems. Beginner systems tend to be more difficult to maintain and use than sophisticated systems. Also, response time for the end user at his/her terminal can be worse for poorly constructed systems that do the same things as sophisticated systems.

When you are building systems in SAS/AF, the goal you shonld be striving for is to build systems with the following characteristics:

- Easy for end nser to use
- Flexibility - Use parameters
- Easy Maintenance.

Although some people might interpret these objectives as implying that systems should be kept simple, that is not the case. A programmer might have a very sophisticated task to perform in order to keep systems simple for the end user .Also, easy maintenance does not imply simple code. Instead, it means designing your system so that maintenance tasks are as menu driven and automated as possible. The code for doing this is not always beginner level code.

Every software product has some enthusiasts who go overboard. They think that their favorite software is so great that it eliminates the usefulness of other software products. Some people might try to tell you that SAS/AF eliminates the need to use ISPF® panels and SAS macros. Nothing could be further from the truth! While SAS/AF does give you a full screen menu system and conditional execution of SAS code, it does not give you the full power of either macros or ISPF. These products should be used to complement SAS/AF and are not replaced by it. In fact, SAS/AF is designed to work in conjunction with SAS Macros. Technical Report P-146 from SAS Institute has an example of this. If you are working outside of SAS, then it often makes more sense to use ISPF menus. Within SAS, SAS/DMI™ is available for accessing ISPF. Each of the products has its particular strong points. Neither is a replacement for the other even though many of their functions overlap.

Procedures Used in SAS/AF

The procedures used in SAS/AF are Proc Build, Proc Display, Proc Catout, Proc Cport and Proc Cimport.

Proc Build is used for creating, modifying, copying, and printing SAS/AF screens.

Proc Display is used for displaying screens created with Proc Build. Proc Display is the procedure that generates SAS code based on what an end user types into the terminal.

Proc Catout is used to generate either a SAS data set or a sequential file containing the directory information for a SAS catalog. This can be useful for keeping track of which screens are in particular catalogs.

Proc Cport and Proc Cimport are used for moving SAS/AF catalogs from one operating system to another.

How Does SAS/AF Work?

SAS/AF generates SAS code based on what an end user types on the screen. This is what allows non-programmers to run SAS code without understanding the SAS system. Figure 1 shows how one Proc Display statement generates SAS code. This code has anther Proc Display which, generates more SAS code. Usually, one "initial" Proc Display will generate all of the SAS code for an end users' SAS session.

Different Types of SAS/AF Screens

The different types of screens available in SAS/AF are Menu, Program, Help, List, Attribute, CBT, Form, and Keys screens. These screens are used together to build your systems. CBT, Form, and Keys screens are not discussed in this paper.

The uses of the other screens are summarized below:

Menu Screens Uses

Menu screens are used to select a task or group of tasks to perform. They do this by calling a program screen or another menu screen. (CBT screens can also be called.) Menu screens make setting up SAS/AF systems relatively easy. They are especially great when you are just learning how to use SAS/AF. But as you gain experience, you will probably use Program Screens in places where you would have used Menu screens when you were a beginner.

Program Screen Uses

Program screens are the heart of SAS/AF. They are the most flexible and powerful of the screens. In fact, you could build entire SAS/AF systems with just program screens. I do not recommend that approach, but I mention it to give you an idea of how versatile program screens are. Also, program screens are the only SAS/AF screens that generate SAS code. As you become more proficient with SAS/AF, you will discover more and more uses for Program screens. You will tend to use program screens more often in situations where you might have previously used Menu screens. A list of some important uses of program screens is shown below:

- Validate User Inputs
- Act as Menus
- Generate SAS® code
- Write to a System file
- Error Messages

An example of a program screen used as a primary menu is shown in figure 2. The advantage of using a program screen like this is that you can tell your system to perform all tasks at one. This saves you the time and trouble of having to individually request each task on the primary menu.

Figure 3 is a menu screen. Figure 4 is a program screen. They perform the same function and illustrate another way that program screens can be used as menu screens.

Help Screen

Help screens provide very useful on-line documentation for your screens. Program screens should have help screens if there could be any question in a users' minds concerning what they are supposed to do. Sometimes, putting detailed instructions on the program screen makes it look too cluttered and difficult to understand. That is when Help screens can be very useful in your SAS/AF systems.

List Screen Uses:

- Long list of valid values
- Standardization of valid values

List screens are accessed by program screens. So if more than one program needs the same list of values (even if the list is really only one value), then you should have your program screens refer to the List screen. If your program screens refer to list screens, you only have to change your values in one place rather than in several if your list of valid values changes.

Also, if you have a list of valid values in a system file, you can read these directly into a List screen without having to retype them. This is documented in *Technical Report P-146*. Doing this can save someone a lot of typing.

Attribute Screen

Attribute screens do the behind the scene work in SAS/AF screens. In program screens, they are used to determine how user input fields are displayed, initialized, supported by on-line help screens, and validated.

The function of a Menu screen is totally determined by the attribute screen. What you see in the text part of the screen is for cosmetic purposes only. It is possible to have "hidden" options that can only be discovered by looking at the attribute screen because they are not shown in the text portion of the screen.

Useful Techniques for SAS/AF™

Here are a few useful techniques to try when you are using SAS/AF. Most of these techniques are useful for SAS in general rather than only for SAS/AF in particular.

AUTOEXEC Command

Display Manager commands can be stored in a file with the fileref of SASEXEC. When SAS is invoked, AUTOEXEC commands, in the SASEXEC file, are executed.

An example of a set of Autoexec commands you may want to use is shown below:

```
%LET AFLIB = FORMAT;

%LET AFCAT = FORMAT;

SUB 'PROC DISPLAY C =
&AFLIB..&AFCAT..DEFAULT.PROGRAM; RUN;';
```

Sysparm Option

This SAS option is used to send information from the operating system to SAS® System

In this example, the value of &DEBUG determines whether extra output will be generated by SAS programs in order to aid in debugging the SAS code.

```
SAS OPTIONS(SYSPARM='&DEBUG')
```

TSO CLIST for Running Your System

In your TSO CLIST, use keyword parameters to control whether you get extra output useful for debugging your code. DEBUG is used to generate extra output from your SAS programs. TRACE is used to generate extra output to your screen from the TSO CLIST that runs SAS. Also notice that a SASEXEC file is allocated in order to use the AUTOEXEC command.

```
PROC 0 DEBUG TRACE

CONTROL NOMSG

IF &TRACE=TRACE THEN +

    CONTROL CONLIST MSG LIST

ALLOC FI(SASEXEC) +

    DA('SYS1.SASEXEC(FORMATS)') SHR

ALLOC FI(FORMATS) +
    DA('SYSTEM.FORMATS.SASAF') SHR;

SAS OPTIONS(MAUTOSOURCE DMS S=72 +
    SYSPARM='&DEBUG') &TRACE

FREE FI(SASEXEC FORMATS)
```

Use SAS/FSP® Products to Look at Data:

Since SAS/AF operates in a full screen environment, you should take advantage of SAS/FSP products for data entry and displaying data and reports. If used properly, they can make your systems easier to maintain and more respousive to your end users.
The following list summarizes some uses of the SAS/FSP procedures.

- FSEDIT - Examine & Modify SAS® data sets
- FSBROWSE - Examine SAS® data sets in FSEDIT format
- FSPRINT - Examine SAS® data sets in Tabular format
- FSCALC - Examine & Modify data in a Spreadsheet format.
  - Can use SAS® data sets
- FSLIST - Examine text files (not SAS® data sets)
  - Look at Reports.

FSEDIT - Helpful Hint

This will prevent end users from accidentally altering your FSEDIT screen. Also, it allows several users to use the same screen if they are editing different data sets.

Store data and screens on separate System files
E.g. data: data.dataset
screen: screen.datascrn

Give users Read/Write access to the data set to be edited (disp=old) and Read only access to the FSEDIT screen (disp=shr).

Maintenance Sub-Systems in SAS/AF

Automatiug maintenance functions is useful to do in SAS/AF. This relates back to making the programmer's job easier and making the results more reliable. If there are functions that should be performed to properly maintain the system you have created, setting up options to do the work will make your job easier. That is really important for activities like generating documentation. If it is easy to do, it is more likely that it will be done. It is also more likely that maiutenance activities performed in a menu system will be done correctly.

Some examples of useful options for a maintenance screen are listed below.

- Change SAS/AF™ screens with Proc Build
- Use Proc Datasets on system datasets
- Generate Documentation
- Data set modification - especially FSEDIT Screens

An example of code you would execute to change your SAS/AF screens from inside your system is shown in figure 5. This would typically be stored in a screen called build.program.

Documentation Techniques

The documentation of concern here is technical documentation for system maintenance. User documentation is important, but is not discussed in this paper. There are four documentation tools that are useful for SAS/AF. They are the AFLIST Macro, Charts and Diagrams, PROC CATOUT, and PROC BUILD.

The AFLIST Macro provides comprehensive documentation for SAS/AF screen catalogs. It is the best software currently available for documenting SAS/AF screens. All information needed by a programmer to understand a system and maintain it are is provided by AFLIST.

Charts and Diagrams can provide an overview of a system more effectively than program listings or AFLIST can. They are great for understanding data flows and interactions between different parts of your system. Figure 6 is an example of documentation for a SAS/AF menu system. Each bubble represents an AF screen. The lower number is the name of the screen. The screen type (e.g., program or menu) is omitted because each screen name is unique. The number on top is what the user types to get to that screen from the main menu which is numbered 0. For example, to get to screen 8.8 from the main menu, the user would type "8" at the primary menu and "8" at menu 8. This numbering system made it easier for end users to identify screens. Remembering a few numbers is often easier than remembering long screen descriptions. This is especially true if the numbers correspond to the key strokes a person must type to use the screens.

PROC CATOUT is used to put a directory of SAS catalog into a SAS dataset or a sequential text file. CATOUT is useful for keeping track of which screens are in which catalog. It does not provide good information about how the screens work.

PROC BUILD can be used to print the contents of SAS catalogs, but it is not as useful as AFLIST. PROC BUILD does not print attribute screens which is a critical deficiency. Also, PROC BUILD does not provide an index of screens for the printout and a table showing which program screens use each macro variable

Your best documentation tools are AFLIST, diagrams, and documentation such as a user guide which describes how to use the system.

Examples

Figures 7 and 8 contain examples of code in program screens that could be helpful to you.

Figure 7 shows you how to print a report to your system printer. This is a simple, but effective method. Simply use proc printto to direct output to a file allocated to your system printer.

Figure 8 is an example of a program that writes records to a file. The records are a job stream for a batch job. Next, the job is submitted and the file, which is no longer needed, is deleted. The lines between "=== batchjob" and "===" are written to the batchjob file.

Conclusion

Now that you have the tools and insights to start using SAS/AF effectively, you can begin to build user friendly systems for yourself and for end users who do not know the SAS system. The only way you will really learn SAS/AF is by building some systems on your own. If you build simple applications, you will get immediate benefits from being able to build user friendly screens quickly. This is the best way to start learning SAS/AF.

If you plan to develop major systems, you must be willing to try new, innovative techniques, and learn lessons from systems you and others have previously built. You should check the SUGI Proceedings for papers that demonstrate programming techniques you need.

SAS/AF will enable you and your co-workers to perform many tasks more easily and more reliably.

References

From SAS® Institute Inc.:

SAS/AF™ User's Guide
Technical Report: P-141
Technical Report: P-146

SUGI 12 Proceedings:
James F. Sattler, "Building A Clinical Review System Using SAS/AF™ Software," Pp. 1230-1236.

Katie Hubbell, "Making SAS® Version 5 Work For You: The Design of a Dynamic Reporting System for all Users," Pp. 1237-1240.

Annette T. Harris, "Techniques for Using SAS/AF™ Software more Efficiently," Pp. 631-634.

R. David Sisemore, "%AFLIST - A Macro to Document SAS/AF™ Software Libraries," Pp. 1220-1225

Feel free to contact the author at

Levine Software Systems
1770 Bryant Avenue S. #416
Minneapolis, MN 55403
(612) 377-5714

Figure 1

Stacking SAS® Statements

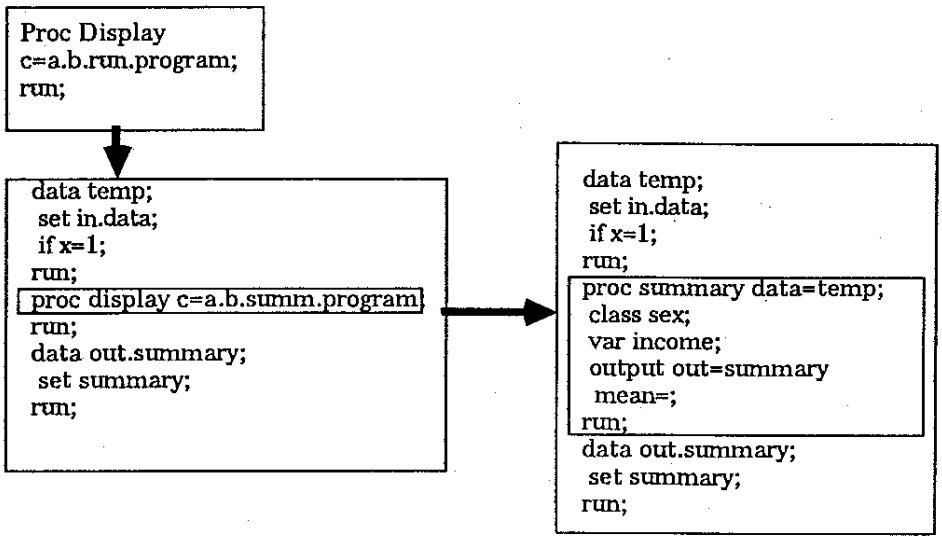

Figure 2

```
Format Editing System

          Select a Format to Work on:
           &format

           Select what you want to do:

           & Sort by right side (left is default)
           & Edit
           & Browse
           & Print

     & Exit System


     HELP:  For Information about any of the fields, place the
            cursor on the field you want to know more about.
            Then press PF1.

            For General Information about this screen, press PF1
            anywhere else.

            Type "quit" on the command line to exit screen if
            all else fails.

  & Use Maintenance Screens
&quit &dms
--------------------------------------------------------------------------------
### MPROG
 %MACRO MPROG;
```

Figure 3

```
                    Maintenance Options:

                    1. Edit SAS/AF Screens with PROC BUILD
                    2. Print SAS/AF Screen Documentation with %AFLIST
                    3. Create a new Format
                    4. Look at format code generated by the system.

                    U. User - Return to User Screens
                    X. Exit to TSO


MASTER HELP SCREEN :
PARENT SCREEN     NAME                          LIBREF    *
                  TYPE                          CATALOG   *
              OPTION    KEY    NAME       TYPE        LIBREF     CATALOG
                1         __   BUILD      PROGRAM     *          *
                2         __   PRINTDOC    PROGRAM     *          *
                U         __   PRIMARY     PROGRAM     *          *
                X         __   EXIT        PROGRAM     *          *
                3         __   NEWFMT      PROGRAM     *          *
                4         __   FMTCODE     PROGRAM     *          *
```

Figure 4

```
          Select a Maintenance Option:

          &A  Edit SAS/AF Screens with PROC BUILD
          &B  Print SAS/AF Screen Documentation with %AFLIST
          &C  Create a new Format
          &D  Look at format code generated by the system.

          &U  User - Return to User Screens
          &X  Exit to TSO
```

```
### MAINTMAC
 %MACRO MAINTMAC;
   %GLOBAL RT_SCRN1 RT_SCRN2;
   %IF &_DCALL = INITIAL %THEN %DO;
       %LET RT_SCRN1 = &LIBREF..&CAT..MAINT.PROGRAM;
       %LET RT_SCRN2 = ;
   %END;
%MEND;
###
>>A BUILD.PROGRAM
>>B PRINTDOC.PROGRAM
>>C NEWFMT.PROGRAM
>>D FMTCODE.PROGRAM
>>U PRIMARY.PROGRAM
>>X EXIT.PROGRAM
```

Figure 5

```
X FREE F(&LIBREF);
X ALLOC F(&LIBREF) DA('UXCN2A.FORMAT.AF.SASDATAB') OLD;
PROC BUILD C= &LIBREF..&CAT ; RUN;
X FREE F(&LIBREF);
X ALLOC F(&LIBREF) DA('UXCN2A.FORMAT.AF.SASDATAB') SHR;
PROC DISPLAY;RUN;
```

Figure 6

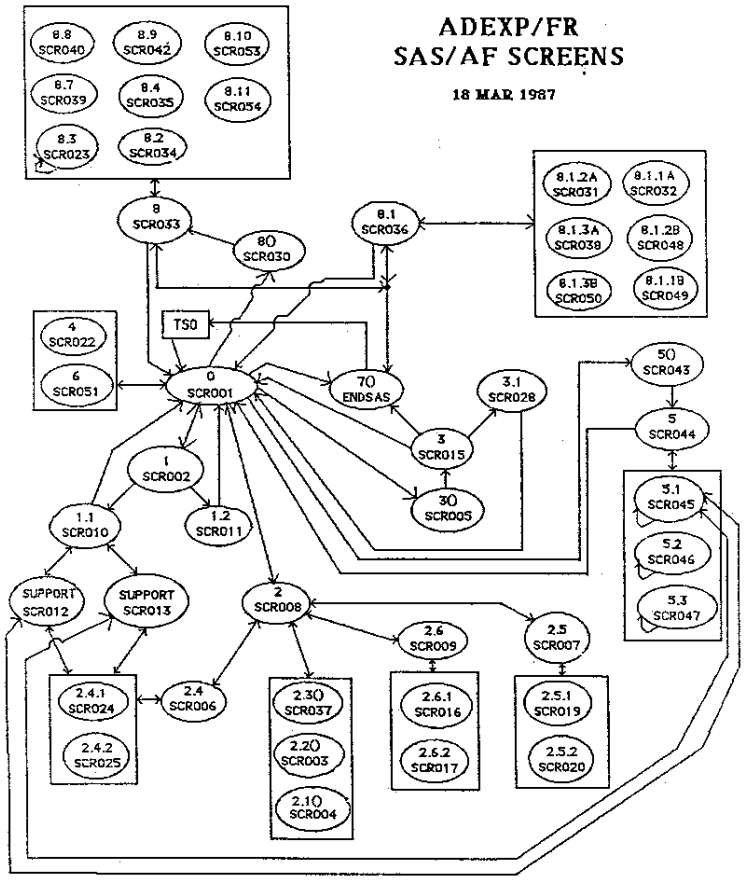

Figure 7

```
##field5
  x free f(ft20f001);
  x alloc f(ft20f001) sysout(h);
  proc printto unit=20 new; run;
  proc print data=formats.&format ;
       title 'The &format Format';
       #field2
         title2 'Sorted by Right Side';
       ~#field2
         title2 'Sorted by Left Side';
##field5
  run;
  x free f(ft20f001);
##

##field6
 ~#field7
  PROC DISPLAY C=&LIBREF..&CAT..EXIT.PROGRAM;RUN;
##

~##field6
 ~#field7
  %menus
##

##field7
  %maint
##
```

Figure 8

```
%MACRO NEWSUB;
 %IF &_DCALL = INITIAL %THEN %DO;
  %TSO FREE F(BATCHJOB);
  %TSO DELETE JCL.MISMATCH;
  %TSO ALLOC F(BATCHJOB) DA(JCL.MISMATCH) NEW TRACK SP(1,1)
       RECFM(F B) LRECL(80) BLKSIZE(6160);
 %END;
%MEND NEWSUB;
###
=== BATCHJOB
//&USERID&FIELD1  JOB (000X,PEX0),'&USERNAME &USERWNDO',CLASS=A,
//     MSGCLASS=T,REGION=3072K,NOTIFY=&USERID
//*
//*******************************************************************
//*         USING SAS FORMATS                                       *
//*******************************************************************
//STEP05   EXEC CMSAS,
//    SYSOUTC=T,WORK=8000,RTNCODE=4095,SORT=5000
//CANC01   DD DSN=&UPDTDSN,DISP=OLD
//CANCEL   DD DSN=&CNCLDATA,DISP=OLD
//HOW      DD DSN=  HOWARD.HOWARD ,DISP=SHR
//SYSIN     DD  DSN=&QLFR2.PROG.SAS(CANCADD),DISP=SHR
===
%TSO FREE F(BATCHJOB);
##FIELD1
%TSO SUBMIT JCL.MISMATCH;
##
%TSO DELETE JCL.MISMATCH;
* PROC DISPLAY C=&DDNAME..&CATALOG..NEWOBS.PROGRAM; RUN;
```