Getting Started with SAS/AF® Software
Howard Levine, Levine Software Systems

Introduction

SAS/AF® is a very effective tool for making user friendly SAS® applications. The purpose of this paper is to acquaint you with some ideas that will help you when you begin using SAS/AF. This paper covers creating applications, but does not cover computer based training (CBT) entries.

The topics that are covered in this paper are ideas to think about when using SAS/AF: what it can do, catalog entry types, the types of screens, 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, enter a selection, or press a function key. By doing this, a user can obtain additional information, alter SAS data sets, change macro variable values, generate or submit SAS statements, or display another screen. 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.

It should be noted that Version 6 SAS/AF differs significantly from its Version 5 counterpart. It has far more capabilities because of Screen Control Language (SCL). SCL is a SAS-like language that permits extensive manipulation of user fields, SAS data sets, and macro variables.

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. Also, there are many ways to validate data that range from simple to sophisticated.

Initially, you may choose simple coding techniques over sophisticated ones. When you are getting started with SAS/AF, you should probably do this. However, make sure that you are prepared to use more advanced methods as your experience and expertise grow. Avoid getting into a rut and establishing standards that lock you into a beginner level.

One example of a "beginner" standard is to require a primary menu for a system to be a MENU entry. As you become more capable, you will want to be able to use PROGRAM entries also. However, PROGRAM entries are often not appropriate for beginners, when speed is more important than elegance, or when the structure and capabilities of a MENU entry are desired for a particular application. Therefore, an appropriate standard would be requiring that the entry initially displayed from a catalog be called PRIMARY, but the programmer should be allowed to decide if it will be a CBT, PROGRAM or MENU entry. This is a reasonable standard because it makes maintenance easier (people know where to start looking at the catalog) without depriving the programmer of tools that are provided by SAS/AF.

With SAS/AF, the goal you should be striving for is to build systems with the following characteristics:

- Easy for end user 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.

**Using SAS/AF**

The only procedure used in SAS/AF is Proc Build. It is used for creating, copying, and documenting catalogs containing AF entries. Other procedures such as CATALOG, DATASETS, CPORT, and CIMPORT are used for maintaining catalogs and transporting them, but they are not essential for building applications with SAS/AF. Proc SPELL can be used for spell-checking HELP and CBT entries. You should examine the documentation for PROC BUILD in the SAS/AF Users' Guide.

**Catalog Entries**

There are several types of catalog entries used with SAS/AF: Menu, Program, Help, List, Key, CBT, and Form. These entries each have various screens that they are composed of such as Display, Attribute (attr), General Attribute (gattr), and Source. The Display screen is used by Menu, Program, Help, and CBT entries; it is the screen you first see when entering the entry in edit or browse mode.

The menu entry is used for providing lists of options to the user. It is very useful for organizing your applications into logical units. Menu entries are very easy to build and understand. A menu entry may call another menu entry, a program entry, a CBT entry, or a help entry. Menu entries use attr and gattr screens. Menu entries are used to select a task or group of tasks to perform. They do this by calling a program entry or another menu entry. (CBT entries can also be called.) Menu entries 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 entries in places where you would have used Menu screens when you were a beginner. However, Menu entries are so simple and easy to set up that you will find them convenient in many instances even as your experience grows.

Program entries are used for a variety of purposes. They are truly the "work horse" of SAS/AF. It is possible to build entire systems with just program entries, but this is often not convenient. It does illustrate that program entries are the most powerful and versatile entries in SAS/AF. Program screens do the "work" of SAS/AF. They are used for verifying user inputs, reading and modifying SAS data sets, submitting SAS code, performing table look-ups, and performing computations. They can also be used as menus which cause other entries to be displayed. Program entries use compiled code. It is important to compile the SCL in the source screen every time a change is made to the SCL or to one of the AF fields. The program entry uses the attr, gattr, and source screens. Program entries are the only SAS/AF entries that generate SAS code and access SAS data sets. As you become more proficient with SAS/AF, you will discover more and more uses for Program entries. You will tend to use program entries more often in situations where you might have previously used Menu entries. A list of some important uses of program entries is shown below:

- Validate User Inputs
- Act as Menus
- Generate SAS® code
- Act as on-line information sources
- Error Messages

One reason for using program entries as menus is that there is much greater flexibility with how the menu is structured. This includes allowing several choices to be picked simultaneously. Also, it is possible to make your menus dynamic by using Extended Tables. In some cases, using a program entry instead of a menu entry as a menu permits the same "work" to be done with fewer catalog entries. This occurs because one program entry can handle all the processing that would have otherwise required a menu entry and several program entries.

Help entries are used to provide information. They can be set up to be displayed from a program or menu screen when a user presses their HELP key. This allows the user to obtain information about a particular field on a program screen or general information about a menu or program screen very easily. Help entries can also be displayed by program entries with SCL or by menu entries as a
Help entries provide very useful on-line documentation for your screens. Program entries should have help screens if there could be any question in a user's mind 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 entries can be very useful in your SAS/AF systems. It is possible to provide help for an entire screen and for each field of a program screen.

List entries are lists of valid values that a program screen field may have. This is one way to standardize lists of acceptable entries for any number of fields. It is also useful if one field has a substantial number of legitimate values. The list of acceptable values can be typed into the LIST entry, read in from a system file, or both.

**List Entry Uses:**
- Long list of valid values
- Standardization of valid values

List entries are accessed by program entries. 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 entries refer to the List entry. If your program entries refer to list entries, 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 entry without having to retype them.

An alternative to list entries is using SAS formats and the PUTC and PUTN functions or the LOCATEC and LOCATEN functions within SCL. These alternatives are preferable if you want to keep your systems dynamic and easy to maintain. But, when you are starting out, it may be simpler for you to minimize the SCL you write. In the long run, avoiding list entries and using the functions with SCL will be very useful. The LOCATEN and LOCATEC functions provide the most flexibility of all data verification techniques and are the most complicated to use.

Key entries allow the developer to customize the keys for a particular screen. This permits function keys to be used exclusively without the user entering a value on the command line or screen. User defined commands can be assigned a key also. This feature is not especially recommended when you are starting to use SAS/AF, but it may be necessary in some cases. It can be very useful if you define your own commands in program entries.

CBT entries are used to provide computer based training. They will not be discussed in any detail in this paper.

Form entries are used to define how catalogs are to be formatted when they are printed. A form should be set up for each print device that you intend to use.

In order to use AF screens, the AF display manager command is used. This is part of the base product. An example of how the command works is

```
af c=mylib.afcat.primary.menu 
```

This will display the primary menu entry of the mylib.afcat catalog. The "af" at the end means that the AF window is to be the active window. Other options to consider are AUTOSAVE=YES, AUTORECALL=YES, and CHECK=YES.

**How Does SAS/AF Work?**

SAS/AF displays screens which display other screens or run SCL programs. The SCL programs are used for verifying user entries into fields on the screen, updating the screen, modifying SAS data sets, and submitting SAS code. AF screen displays are initiated with the display manager AF command. When using SAS/AF, streams and hierarchies are created. These are explained in the Users' Guide and you should understand them. They explain why your application will display a particular screen after another screen has finished its processing.

**Different Types of SAS/AF Screens**

The different types of screens available in SAS/AF are Attribute, General Attribute,
Source, Message, and Preview screens. These screens are used together to build your systems.

The uses of the other screens are summarized below:

**Attribute (attr) Screen**

Attribute screens do 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.

**General Attribute (gattr) Screen**

General Attribute screens are used to set attributes for the entire entry rather than for just one special part of the entry. This controls the "parent screen," screen size and position, general help screen, and other attributes.

**Source Screen**

Source screens are for writing and storing SCL. They are only used in PROGRAM entries and SAS/FSP products.

**Message (msg) Screen**

Message screens are where SCL compile error messages are written. They are accessed when editing a PROGRAM entry.

**Preview Screen**

Stores SCL code that PROGRAM entries generate. You can look at the code and save it to a system file. This is created when you are executing a SAS/AF program screen using the AF command. It is not created when Proc BUILD is being run.

---

### Critical Statements and Functions in SCL with SAS/AF

- **SUBMIT** Starts a submit block. Many options can be specified. IMMEDIATE is a sensible option in many cases.
- **ENDSUBMIT** Ends a submit block.
- **REPLACE** Replaces a non-blank AF field with a user defined string. This is similar to the ## and # symbols used in Version 5, but is better. Can be set on attribute screen also.
- **DISPLAY** Displays another screen.
- **WREGION** Defines the window position for the next window.
- **CONTROL** override default actions by labelled blocks of SCL code
- **DIRLIST** Displays list of data sets in directory.
- **FSEDIT** Edits a SAS data set using PROC FSEDIT.

### Useful Commands

- **TRACE** Turns a program tracing facility on or off
- **SAVE** Save field entries
- **RECALL** Recall field entries used last time
- **TESTAF** Use in PROC BUILD to test screens without submitting SAS code

### 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.
AUIOEXEC Command

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

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

%LET AFLIB = FORMAT;
%LET AFCAT = FORMAT;
DM "AF C = &AFLIB .. &AFCAT .. PRIMARY.PROGRAM" AF

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')

In some operating systems that do not support the SYSPARM options, you could use system variables to get the same thing accomplished.

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(SYSI.SASEXEC(formats)) SHR
ALLOC FI(FORMATS)
DA(SYSTEM. FORMATS.SASAP) 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 responsive to your end users. There are SCL functions as well as SAS procedures that let you run these FSP procedures.

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/FSVIEW - Examine SAS® data sets in Tabular format
• 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

Automating 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 maintenance 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 FEDIT Screens

An example of code you would execute to change your SAS/AF screens from inside your system is shown in figure 1. 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 two documentation tools that are useful for SAS/AF. They are Charts and Diagrams and PROC BUILD.

Charts and Diagrams can provide an overview of a system more effectively than PROC BUILD can. They are great for understanding data flows and interactions between different parts of your system. Figure 4 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 BUILD can be used to print the contents of SAS/AF entries in catalogs. It will print out the display, source, attribute, and general attribute screens for each entry. There are also features designed to show how menu entries are structured in your system. This is not a very sophisticated documentation feature. It does not handle PROGRAM entries acting as menus. It is also fairly clumsy to use. I have written my own software to accomplish the same task more effectively.

The best documentation can be obtained by using a listing from PROC BUILD, diagrams, and written documentation such as a brief user guide. HELP and CBT entries can serve as online documentation.

Caution:

When using macros or macro variable in SCL, remember that they will not change when you redefine your macro or macro variable. You must also recompile the SCL. Use SYMGET and SYMGETN to avoid this problem with macro variables.

Examples

Figure 1 a program that submits PROC BUILD. This allows you edit your screens by selecting from a menu.

Figure 2 is a simple example of an extended table. It retrieves observations with a value of 1 specified by the user.
Figure 3 is a simple screen that adds 5 to what
the user enters. Notice the macro definition.
Remember, changing the macro will not have an
effect until the SCL is recompiled.

Figure 4 is a screen flow diagram.

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 (Version 5, 6, and
6.06 editions)
Technical Report: P-141 (Version 5)
Technical Report: P-146 (Version 5)

SUGI 15 Proceedings:
David Septoff, "Screen Control Language -
An Invaluable Tool in Applications

SAS and SAS/FSP, SAS/AF and SAS/DMI are the
registered trademarks of SAS Institute Inc.,
Cary, NC, USA.
ISPF is the registered trademark of
International Business Machines Corporation,
Armonk, NY.
Figure 1

ENTRY: BUILD.PROGRAM  BUILD.PROGRAM  Last updated: 09/14/90

***** ATTRIBUTES *****
Window name: 
Start row: 0  col: 0
Number of rows: 0  col: 0
General help: Default lookup dataset: Extended table?
Function keys: Parent:

***** SOURCE *****
Term: submit terminate;
proc build =0.cfost:run;
endtermity;
return;

Figure 2 - Use Extended Table
ENTRY: TEST1.PROGRAM  TEST3.PROGRAM  Last updated: 09/12/90

***** ATTRIBUTES *****
Window name: 
Start row: 0  col: 0
Number of rows: 0  col: 0
General help: Default lookup dataset:
Function keys: Extended table: X
Parent:

Field name: II  Pages: 1  Rows: 2  Col: 34  Length: Protect.
Alias: II  Capt: X  Pad: Protect.
Type: NUM  Correct: Just: L  Required:
Format: Autosh: X  Non-display:
Inform: Error color: RED  att: REVERSE  Help: AL  EEL
List:
Initial:
Replace:

Field name: JI  Pages: 2  Rows: 1  Col: 4  Length: Protect.
Alias: J1  Capt: X  Pad: Protect.
Type: NUM  Correct: Just: L  Required:
Format: Autosh: X  Non-display:
Inform: Error color: RED  att: REVERSE  Help:
List:
Initial:
Replace:

Field name: EJ  Pages: 2  Rows: 1  Col: 16  Length: Protect.
Alias: E1  Capt: X  Pad: Protect.
Type: NUM  Correct: Just: L  Required:
Format: Autosh: X  Non-display:
Inform: Error color: RED  att: REVERSE  Help:
List:
Initial:

Figures Containing Examples

Figure 3 - Use Macro to do Calculation
ENTRY: TEST1.PROGRAM  TEST3.PROGRAM  Last updated: 09/12/90

***** ATTRIBUTES *****
Window name: 
Start row: 0  col: 0
Number of rows: 0  col: 0
General help: Default lookup dataset:
Function keys: Extended table: X
Parent:

Field name: RMBDA  Pages: 1  Rows: 1  Col: 20  Length: Protect.
Alias: RMB  Capt: X  Pad: Protect.
Type: NUM  Correct: Just: L  Required:
Format: Autosh: X  Non-display:
Inform: Error color: RED  att: REVERSE  Help:
List:
Initial:
Replace:

Field name: RMBDB  Pages: 1  Rows: 1  Col: 20  Length: Protect.
Alias: RMB  Capt: X  Pad: Protect.
Type: NUM  Correct: Just: L  Required:
Format: Autosh: X  Non-display:
Inform: Error color: RED  att: REVERSE  Help:
List:
Initial:
Replace:

***** SOURCE *****
L:exit

Figure 4 - Use General Macro
ENTRY: PRIORITY.AMG  PRIORITY.AMG  Last updated: 09/12/90

***** ATTRIBUTES *****
Window name: 
Start row: 0  col: 0
Number of rows: 0  col: 0
General help: Default lookup dataset:
Function keys: Extended table: X
Parent:

Field name: RMBDA  Pages: 1  Rows: 1  Col: 20  Length: Protect.
Alias: RMB  Capt: X  Pad: Protect.
Type: NUM  Correct: Just: L  Required:
Format: Autosh: X  Non-display:
Inform: Error color: RED  att: REVERSE  Help:
List:
Initial:
Replace: