Designing Complex Applications
Using SAS/AF Screen Control Language

Eric Klusman
Federal Reserve Bank of Chicago

Introduction.

This paper discusses certain issues related to the design of complex applications using the SAS system, version 6.03, under MS-DOS on IBM-compatible personal computers. Now that the full SAS System is available on PCs, our department is moving from mainframe-based SAS 5.18 to SAS 6.03 on a PC Token Ring Local Area Network. By using examples and a summary, this paper will discuss the question of when in a LAN environment to design a pure SAS application, using just the tools of the SAS System, especially SAS/AF Application Facility and its new Screen Control Language (SCL), versus when to design a mixed application, using tools from outside the SAS System.

Previous environment: mainframe.

Until recently, the department’s automation focused on complex applications on a large IBM mainframe computer running TSO under OS/MVS/3A, with PCs used mostly for word processing and terminal emulation. These mainframe applications featured concurrent execution of the SAS System and ISPF, IBM’s dialog management and panel facility, in addition to other applications software. These applications comprise 15,000 lines of installation-written code to provide data analysis, production report-writing, and maintenance and documentation facilities. Because these applications greatly increased productivity in the department, our challenge was to exploit new technologies, like Local Area Networks, while keeping the advantages of complex applications. Obviously, new tools allow new solutions; our goal is to increase productivity, not exactly duplicate mainframe applications on PCs.

Current environment: PC LAN

The Research Department of the Federal Reserve Bank of Chicago conducts research on regional, national, and international economic issues and prepares publications, policy briefings, and other presentation materials.

The main users of the LAN are the research economists, the technical support staff that manages the hardware, software and data used by economists, the publications production area, and the managers of the department and their assistants. This paper focuses mainly on the research economists’ applications needs.

The LAN is used for many standard functions, including data analysis, using the SAS System, version 6.03 for PCs, and other software; word processing; electronic mail; personal calendars and group scheduling; and certain management and security functions, such as backups, previously provided for the department by the central mainframe staff.

These functions are implemented on a Novell SPT NetWare SPT network configured as Token Ring. Servers and the typical workstation are 80386-based PCs. Servers provide approximately two gigabytes of disk storage, while the typical workstation, running MS-DOS, has a forty megabyte hard disk. Other resources on the LAN include mainframe and asynchronous communications gateways and a CD-ROM data server.

Pure versus mixed applications.

For purposes of this paper, an application is a sequence of these operations: get information from the user, verify the information, act on the information, and present the results to the user.

A pure SAS application is written using only the tools available in the SAS System. This is, of course, an extremely powerful set of tools covering an extremely wide degree of function. When developing applications in the SAS System allows fairly rapid prototyping, quick development (since so many tools are ready-made), and, especially with SAS/AF, easy
documentation. Moreover, compared to other approaches, most SAS applications are significantly easier to maintain (SAS is easier for a new staffer to learn than, for example, C).

On the other hand, an mixed application is written using the tools of the SAS System as well as other software tools. Mixed designs often arise when application requirements are many in number or interactive, then mixed designs may be the only way to deliver performance. By custom-developing specialized code for demanding or frequently-executed applications, the development can meet that increased need to do each part well. Additionally, it may be possible to combine several separate kinds of software, playing to the strengths of each.

Obviously, the extra performance of a mixed application comes at extra cost. The developer is in charge of more details, especially if working in a lower level language, or if fitting many pieces together. Such designs may involve complicated transfers of control between tools (a relatively simple example is offered below). Moreover, the scope of variables may shrink dramatically as well as resist control. The question becomes: what information from the user can I make available where?

An applications developer with some expertise with the SAS System faces another challenge. Some tools previously available in SAS are not available in version 6.03, including user written procedures, statement-style macros, and command-style macros.

SAS A/F Screen Control Language.

One of the best new features of version 6.03 of the SAS System is the Screen Control Language (SCL) of SAS/AF, the applications facility product. With SCL, the developer can more effectively organize larger and complicated applications. SAS/AF has built in tools to test as well as document (including cross-references between modules) such applications.

With SCL the developer has much more control over the screen definition than in the SAS macro language's %WINDOW - %DISPLAY method. SCL will verify a great variety of field types as well as present selection lists of variables, datasets, catalogs, and filenames. The developer's SCL program can contain sections for processing before the panel is displayed (INIT), after each set of user inputs (MAIN), and when the user enters an END or CANCEL command (TERM). Additionally, the SCL program can parse commands entered by the user on the command line, as well as tailor a SAS program with information from the user and submit that program to the SAS supervisor.

Of course, because the SAS System runs on many platforms, applications developed with SAS/AF and SCL offer the advantages of portability and consistency across many platforms.

Example of a mixed application.

Our department has a standing need for programming panel and browser for users writing SAS programs directly. Such an application allows the user to edit and submit SAS programs, and to browse the resulting input, log, and output streams. The idea is to develop an alternative to the SAS Display Manager System's Edit, Log, and Output windows in order to use a more familiar or powerful editor as well as to have structured input, log, and output streams in order to review, print, recapture and resubmit, and save particular ideas, rather than entire undifferentiated streams. Users want SAS usage organized to keep related work together.

Given the extremely frequent use of this application, an alternative mixed design providing greater performance, described below, is preferable. However, to support the extra costs of a mixed design, a pure SAS design is considered first.
The user sees a screen something like this:

Program Development Screen

Command ===>  
Project ===>  
Topic ===>  
Program ===>  

Actions  
Edit ===>  
Submit ===>  
Log ===>  
Output ===>  

FIGURE 1.

The user enters a project name and an optional topic name, which corresponds to a DOS subdirectory named SAS_LIB\project\topic, and a program name. The user also selects actions to perform on this SAS program. Other features that can be included on the screen include a field for a program to copy into the new program, the option to cycle over the other select options until the user cancels the cycle, and an option to display the Browser screen, which can look something like this:

Browser

Command ===>  

? Name | Label | Size (KB) | In | Log | Out

DB  * build database; .4 .6 .9  
REG  * idea from Ellen; .2 .3 .8  
  
FIGURE 2.

The user would select the stream and submission with some action code, say PL on a line to print that program's log. The list can be presented by descending date and time, or by other sorting criteria, such as name or size.

Here's the structure to implement this application. The key idea is to redirect the source, LOG, OUTPUT streams with PROC PRINTTOS automatically generated by the application. The streams are kept on disk, one submission stream per file. Disk is the necessary destination since the Display Manager System doesn't support sharing DOS memory. Writing to disk has the advantage that the user can run large sessions without memory problems. If desired, the same code can write the streams to a VDISK for near-memory speeds.

SAS/AF's job is to present the screen, and then run this SUBMIT IMMEDIATE block:

X 'verify_create_and_edit.BAT';  
%INCLUDE 'submit_file';

When submitted, this transfers control to a DOS .BAT file running in the nested copy DOS invoked by SAS. This .BAT file uses Pascal compiled programs to verify the verify the subdirectory and change to that subdirectory. This is necessary since SAS/AF cannot verify a subdirectory. The .BAT file then invokes the editor, then write SAS statements to a file for the SCL %INCLUDE before exiting DOS and returning to SAS:

PROC PRINTTO ...;  
%INCLUDE user_program;  
PROC FSLIST on log;  
PROC FSLIST on output;

The Browser (which is a separate application in some designs) would use SAS/AF to present the screen with extended table for the submission streams, then transfer control to DOS for at least some of the user-selected actions (delete).

In an application like this, the requirement is for as much speed of execution as possible. But the application
changes environments many times, and some of the environments (SAS/AF) are not doing that much work. For example, SAS/AF is not much help with this application's needs to manipulate DOS files, rather than SAS datasets and variables. SAS/AF, perhaps understandably, is not at its strength in dealing with such operating system specific functions.

One alternative is to use the SAS macro language, which will present a screen. However, this would make the SAS Log more difficult to read and may cause problems if the user writes poor SAS macro language in the program being processed.

The stronger alternative is to make this (a still more) mixed application, by using some computer language (Prolog, Pascal, C, etc.) to write compiled programs to display the screens and transfer control. The structure here would be to wake up the SAS system, immediately X to DOS, display the panel, verify and change directories, invoke the editor, then exit DOS to return to SAS, this time only to execute the user's SAS program. The application would then exit to DOS with an X command immediately to browse results and edit the next program.

This has the advantage of executing quite quickly, since the user has great control over how the application gets each part done and can optimize the parts. But this mixed design has the disadvantages of losing easy access to SAS help functions, losing easy access to SAS/AF functions for prompting and verifying SAS objects (not a problem, perhaps in this instance), as well as being more difficult to write and maintain.

**Example of a pure SAS application.**

Another need in our department is for a graphics panel for the user to produce analytical graphs without writing SAS/GRAPH code directly. (Presentation quality graphs are prepared using other applications.) This application is similar to those in SAS/ASSIST, distributed by the SAS Institute, and is easily specialized for a particular set of user graphics requirements.

An application like this can answer a common need for quick or shop-standard graphs. It can be useful to both novice users who are not trained in SAS/GRAPH, as well as expert users who could write the SAS/GRAPH program directly but who want to save time. Moreover, some Procedures, like GCHART, often require restructuring the data. Of course, the benefit comes from having the application manage the details of SAS/GRAPH, so it should not ask about SAS/GRAPH details directly.

One design, using SAS/AF, is to use SCL to present a screen with fields for the dataset, variables, and ID variables, perhaps for the horizontal axis. The MAIN section of the SCL program uses REPLACE statements to tailor SAS/GRAPH program appropriately.

**Summary of uses of SAS/AF**

SAS/AF and SCL are powerful tools that will handle a great many applications well. They are especially well-suited to applications that follow a cycle of presenting the screen, prompting for and verifying the requested SAS objects (datasets, variables, catalogs), then performing the requested actions on those objects.

In using SAS/AF and SCL, the developer should be aware of certain aspects of flow of control in SCL. A SUBMIT block ends an SCL section, so the developer cannot get additional function from say, base SAS, in the middle of SCL processing. This fits with the SAS Systems general organization into steps and step boundaries. Moreover, CALL EXECUTE happens when control is transferred from the developer's SCL program back to SAS/AF, not immediately within the SCL program. One way to augment SAS/AF inside SCL programs is to pass information via macro variables, which SCL can read and write.
The scope of SCL variables is only with SCL. This forces the developer to macro variables to pass information, as noted, but SAS/AF provides four macro variables always directly in memory for quick access. As an advantage, since the typical user will not be writing SAS/AF programs, chances for inadvertent interactions between the user’s program and the application itself are reduced.

The one obvious small improvement to SAS/AF that would pay big benefits would be to add a function to verify LIBNAME (subdirectory) to complement the function that already exists to verify a DOS file. These benefits would be greatest in MS-DOS, since subdirectories can hold both SAS datasets and SAS programs; in, for example, and OS/MVS environment data libraries and source libraries can never be the same. In any case, many applications may need to manage LIBNAME statements for the user — except that currently, a LIBNAME cannot be verified and if it fails, SAS/AF halts.

Returning to the overall question of pure SAS applications versus mixed applications, it is certainly clear that the SAS System is extremely capable of complexity, with its wide variety of function running on a wide variety of platforms.

A mixed design should be used when the extra costs of doing so are exceeded by the resulting performance gains. Such applications may arise when using data in a format not supported by SAS, such as IDMS, Compustat financial data, or other proprietary formats, or when application requirements are so demanding and so far outside SAS’s turf that the extra work is profitable.

Acknowledgements and disclaimer.

I benefited from discussions on these topics with Steven Strongin, Assistant Vice President, Federal Reserve Bank of Chicago. All errors in this paper, however, are, of course, mine.

The views expressed in this paper are solely mine, and should not be construed as the views of the Federal Reserve Bank of Chicago, its officers, or its directors.

Communicating with the author.

The author may be contacted as follows:

Eric M. Klusman
Federal Reserve Bank of Chicago
Economic Research Department
230 South LaSalle Street
Chicago, Illinois 60604

(voice) (312) 322-5601
(FAX) (312) 322-5700.