SAS® MACROS AND A SAS/AF® MENU SYSTEM FOR READING COMPUSTAT® FILES

Christian Vye, University of Rhode Island
Craig Dickstein, Blue Cross/Blue Shield

OVERVIEW

The University of Rhode Island subscribes to the Standard and Poor's COMPUSTAT® II financial database. The files are arranged on seven standard labelled tapes: 1) the Annual Primary, Secondary and Tertiary files, 2) the Quarterly Primary, Secondary and Tertiary files, 3) the Annual Bank file, 4) the Quarterly Bank file, 5) the Over-the-Counter file, 6) the Industrial Research file, and 7) the Prices-Dividends-Earnings file. Seven SAS® macros are available to read the tapes. The macros generate a SAS data set for each file. Additionally, a SAS/AF® menu system has been provided which creates typical SAS jobs and submits them for Batch processing by the IBM® 4381 mainframe.

JUSTIFICATION

The tapes contain large amounts of data in real binary format, in arrays, and in character fields. SAS, the most widely used mainframe package, has data management capabilities well suited for handling this data. The SAS Macro language, with its Autocall facility, provides a very good way to modularize code and make it available to a broad user base. Providing the macros reduces consulting load dramatically.

The OS/MVS operating system and its Job Control Language (JCL) compounds the difficulties of many researchers and students. The ability of the SAS/AF system to write to external files (see SAS Tech. Report P-146, pp. 144-145; Harris, 1988) provides a way for JCL to be generated for the end user. In addition, SAS code itself can be provided, so that non-SAS users, such as new undergraduate students, can make reports and summarize the data right away.

THE SAS MACROS

The SAS macros created to read the COMPUSTAT data files are based on the %SPYR macro from the SAS/ETS® sample library (member SPYR); see pages 608-609 in the SAS/ETS User's Guide, Version 5 edition.

There is one major difference in the order in which macro statements build the SAS code, as well as a few minor differences that stem from updates to the data file over the years.

To understand the SPYR macro and our local change, let us divide it into 7 parts, A through G. These parts are defined by the comments in the code:

A - this section includes the %MACRO statement and the first INPUT statement, which has the comment 'Input the Company-Level Fields'
B - 'Generate if statement based on selection criteria, if given'
C - 'Read the Data for 20 years (Years per Record)' D - 'Generate an If statement if Begyr and/or Endyr given'
E - 'Process Data for a year if that Year is accepted'
F - 'Generate an END statement if a DO given earlier'
G - 'Generate a KEEP statement if Select was specified' and the %MEND

In our version of the SPYR macro, the order of the parts is A, C, D, E, B, F, and G. Section B was moved after section E so that all variables are defined before they are used in the subsetting IF statement. In the SAS sample version, any variables defined after section A could not be used in the SELCRIT list, because at that point they did not exist in the data set. The original section B from the sample library as shown below:

```sas
%IF %LENGTH(&SELCRIT) = 0 %THEN
   IF &SELCRIT %STR(;);
%END;
%IF %LENGTH(&SELCRIT) = 0
   &LENGTH(&BEGYR)
%THEN %DO;
   IF &BEGYR <= YEAR <= &ENDYR THEN
      IF &SELCRIT THEN OUTPUT;
   %END;
%END;
%IF %LENGTH(&SELCRIT) = 0
   &LENGTH(&BEGYR) = 0
%THEN %DO;
   IF (&BEGYR <= YEAR <= &ENDYR) THEN
      OUTPUT;
   %END;
%END;
%IF %LENGTH(&SELCRIT) = 0
   &LENGTH(&BEGYR) = 0 THEN OUTPUT;
```

The original section E has an OUTPUT statement at the end. In our version the OUTPUT statement was omitted from section E, and inserted in section B:

```sas
%IF %LENGTH(&SELCRIT) = 0 &
   %LENGTH(&BEGYR) = 0
%THEN %DO;
   IF &BEGYR <= YEAR <= &ENDYR THEN
      IF &SELCRIT THEN OUTPUT;
   %END;
%END;
%IF %LENGTH(&SELCRIT) = 0 &
   %LENGTH(&BEGYR) = 0
%THEN %DO;
   IF (&BEGYR <= YEAR <= &ENDYR) THEN
      OUTPUT;
   %END;
%END;
%IF %LENGTH(&SELCRIT) = 0 &
   %LENGTH(&BEGYR) = 0 THEN OUTPUT;
```
THE SAS/AF SYSTEM

The menu system has two purposes: 1) to create an MVS Batch job, and 2) to submit it. The job contains JCL and the appropriate SAS macro call to read the correct tape. SAS statements can be added for further customization before the job is submitted.

The Flow of Control (see Figure 1):

1) Under TSO the user invokes a CLIST, 'COMPSTAT', to begin the appropriate SAS/AF session. The CLIST has a number of functions. It allocates an OS data set to hold the batch program in the user's account. Upon completion of the JCL and macro invocation the user may want the ability to add SAS code to the data set more than once. A home-grown utility called DSRESET makes adding to a new permanent data set possible by changing the disposition (DISP) of the data set from NEW to MOD. Finally the CLIST invokes SAS and executes PROC DISPLAY which brings up the first menu.

2) The first menu screen, ENTRY.MENU, labelled 'Entry Point', presents the user with the following choices:

   1- Produce a brief report (this assumes a small amount of the data will be processed)
   2- Produce a longer report (this gives the user control over many of the JCL parameters that allow a longer job to execute)
   3- Process current jobs (this directs the user to the PROCESS.MENU screen)
   4- Exit from the system
   5- Get help on the current screen
   6- Get a description of the entire application

Choices 3, 5, and 6 return the user to the ENTRY.MENU screen. If option 1 is selected the user gets directed to a second catalog, CATLG2, otherwise the same catalog, CATLG1, is used. The next 3 screens are identical in name and function between the two catalogs, but they differ in content.

3) If choices 1 or 2 are selected, the user goes to the REPORT.MENU screen each catalog. This is the menu where the user selects which Compustat file will be read.

4) The next screen in either catalog is a program screen the user must execute unconditionally. There are 7 of these in each catalog, one for each Compustat file. The program screen invokes the %ALLO (short for allocate) macro, which re-allocates the OS data set to a disposition of OLD.

5) The following screen in either catalog is a program screen named for the file being read (e.g., SPYR.PROGRAM). Within catalogs there are 7 of these, which are similar except for the SAS macro called and screen messages. Between catalogs, the major difference is in the number of choices (active fields) given to the user. In the simpler screens of CATLG2 the same fields exist, but many are protected and not visible.

6) If the program had branched off to CATLG2, PROC DISPLAY will return control to CATLG1. A program screen containing the %ALLO macro call will reallocate the data set with a disposition of MOD.

7) The next screen is a menu screen named for the file being read (e.g., SPYR.MENU). Here the user can enhance the program by choosing:

   1- Add Titles and Footnotes
   2- Put pre-defined Labels in place of the variable names
   3- Give the variables custom Labels
   4- Execute PROC SUMMARY
   5- Execute PROC PRINT
   6- Go to the submit program screen
   7- Return to the Entry Point without submitting

   Choices 1 - 5 return the user back to this screen. The PROC PRINT program screen also has the option to go directly to the submit program screen.

8) Once in the submit program screen the user has a number of choices: 1) submit the batch job by filling an action field, 2) save the program into a permanent OS data set by filling in a data set name field, 3) read another tape by filling an action field, and 4) leave all the fields empty. The IBM add-on utility COPY will allow the user to make a copy of the original data set. If the user had selected choice 3, then return is to the REPORT.MENU screen in the appropriate catalog. Otherwise, the destination from the submit program screen depends on which kind of report the user had chosen to make. For 'Build a brief report' the return is to the Entry Point. If the user had chosen 'Build a longer report' the return is to the PROCESS.MENU screen, which provides the following choices:

   1- Read another tape
   2- Process jobs with ISPF
   3- Exit
   4- Help

This menu retains control in CATLG1, so that 'Build a brief report' is no longer an option. When the user chooses to exit the system, the CLIST ends the SAS session and deletes the OS data set created at the beginning of the CLIST.
Problems and Solutions:

We want to submit the Batch job from within SAS/AF. To do this we need to use the TSO SUBMIT command which requires a permanent OS data set. We allocate the data set in a CLIST and write the SAS/AF-generated statements to the data set by using the === option. But we also want the ability to add more statements to that data set. To do this we must change the data set's DISP from NEW to MOD. For a permanent data set this is only possible after applying the in-house utility, DSRESET. DSRESET writes an end of file marker at the beginning of a permanent sequential data set. Within the SAS/AF application the DISP of the data set is changed by using the TSO ALLOCATE command from within a SAS macro (%ALLO). Placing the TSO command from within a macro accomplishes the following: it allows a TSO command to be executed without user intervention (e.g. carriage returns); the userid, which must be in the data set name, can be obtained from the automatic macrovariable SYSJOBID; the code can be repeated for different DISP and data set name values by using keyword parameters. In the first program screen where %ALLO is used, the DISP is set to OLD. The first time the user gets to this screen this action is unnecessary. However, this allows the user to read another tape in the same session; the data set will be 're-wound'.

SAS Macros solve a variety of problems in this application:

1) Conditional Execution: A very convenient feature of SAS/AF, the conditional execution indicators (e.g., #variable or ##variable) are generally unavailable. This is because most SAS statements are written to an external file by using the === option, and the only code allowed above the === DDname option in the program portion of the program screen is the ### macro (see SAS Tech. Rep. P-146, pp 141-147). For example, PROC PRINT and PROC SUMMARY are written as SAS macros to allow the user to choose the options or statements under each procedure. The user field values are assigned to the keyword parameters in the macro call below the === option.

2) Field Initialization: The ### macro is used to assign values to macrovariables. User fields in the program screens can have macrovariables associated with them via the ATTRibute panel. The macrovariables can be initialized from within the ### macro by using %LET statements in a DO group when the &_DCALL automatic macrovariable's value is INITIAL, e.g.:%

\[
\begin{align*}
&\text{\%IF \&_DCALL = INITIAL \%THEN \%DO;} \\
&\quad \%LET TIME = 1; \\
&\quad \%LET LINES = 2; \\
&\%END;
\end{align*}
\]

The macrovariable value, and with it the field's value, can be updated by the user if a DO group like the one below is also included in the ### macro:

\[
\begin{align*}
&\text{\%IF \&_DCALL = END \%THEN \%DO;} \\
&\quad \%LET TIME = &TIME; \\
&\quad \%LET LINES = &LINES; \\
&\quad \%LET DEST = &DEST; \\
&\quad \%END;
\end{align*}
\]

3) Consistency Checks on User-Fields: A consistency check (see the example in SAS Tech. Rep. P-146 on pp 141-147) is done for an output SAS data set name field (OUTPT) and an OS data set name field (DSN). The issue here is whether the user is storing the data in a temporary or permanent SAS data set. In the case of a permanent SAS data set the OUTPT field must contain a two-level name and the DSN field contain a valid name (preferably the correct name of an existing OS data set in the user's disk space, but there is no check for that). In the case of a temporary SAS data set, the DSN field must be blank and the OUTPT field could be blank or contain a one-level name. The first level in a two-level name in the macrovariable &OUTPT is extracted using the %SCAN and %LEFT functions and put into a macrovariable called &FLVL. Then the lengths of the three macrovariables &OUTPT, &FLVL, and &DSN are compared:

\[
\begin{align*}
&\%IF ((\%LENGTH(&OUTPT) = &DSN =) & & ((\%LENGTH(&OUTPT) = \%LENGTH(&FLVL)) & & & \%THEN \%DO; \\
&\quad \text{Sound alarms, etc....}
\end{align*}
\]

Because a permanent SAS data set is optional, there cannot be a JCL statement for it. Fortunately, the LIBNAME statement is available. A macro called %LIB allocates the DDname (the first level in the two-level name represented by the FLVL non-display field) to the data set entered in the DSN field:

\[
\begin{align*}
&\%LIB(DDNAME=&FLVL,DSNAME=&DSN)
\end{align*}
\]

Within the %LIB macro there is another example of conditional execution of SAS code by using macro language:

\[
\begin{align*}
&\%IF &DSNAME ^= \%THEN \%DO; \\
&\quad \text{LIBNAME &DDNAME "&DSNAME"}
\end{align*}
\]

\[
\begin{align*}
&\text{DISP=OLD;}
\end{align*}
\]

\[
\begin{align*}
&\%END;
\end{align*}
\]
4) Efficiency: The pre-defined labels are in macros because they hold a lot of code. All macros (not counting the ### macros) used in this application are stored in an Autocall library, which 1) allows the program screens to be a more manageable size, 2) cuts down on compilation time for oft-used macros (e.g. %ALLO), and 3) helps the developer to keep track of them. The titles, footnotes, and custom labels were not written into macros because they used relatively few statements and, more importantly, they are statements that can be left blank. For instance, TITLE ; is a valid SAS statement.

5) Screen Communication: Three macrovariables were initialized in Step 5 where the JCL and macro call were added to the data set. The PROC PRINT, PROC SUMMARY and user Labels screens do not have to be repeated because &FILETYPE is used to store the name of the macro used to read the specific Compustat file. Because the enhancement menu screens are named for the macro, return to the appropriate menu was accomplished by the following command:

```
PROCDISPLAY
C=AF.CATLG1.&FILETYPE..MENU;RUN;
```

The same method is used for the submit program screen. The macrovariable &CATLG contains the name of one of the two catalogs. If the user chooses to 'create another program' the catalog to return to is determined by the value of the &CATLG macrovariable. In addition, because the submit program screen does not add code to a data set, it can use the conditional execution indicators. The correct menu to return to by default is determined by the macrovariable &FLAG, which acts like a Boolean operator; it contains a value in one catalog and is left null in the other catalog. The conditional execution indicators utilize fields, not macrovariables, so a field is created (FLAG) in the submit program screen which is associated with &FLAG and made non-display by using the ATTRibute panel. The program portion of the submit program screen demonstrating these methods is shown below:

```
##FIELD8
PROCDISPLAY
C=AF.&CATLG..REPORT.MENU;RUN;
```

```
^##FIELD8
#FLAG
PROCDISPLAY
C=AF.CATLG1.PROCESS.MENU;RUN;
^#FLAG
PROCDISPLAY
C=AF.CATLG1.ENTRY.MENU;RUN;
```

Help is invoked with the F1 key and comes in the form of options from a menu screen. The cursor may be in a program screen field or at the command line. In the latter case, the SAS Institute decided to 'hard code' the destination to be a screen called DISPLAY.HELP. Therefore, a DISPLAY.HELP screen was written for this application that describes the application. If you do not write a DISPLAY.HELP screen, the user will receive an Error message saying that DISPLAY.HELP does not exist.

Considering the extent of this application, there are not a lot of Help screens. Many of the Help screens are shared by similar program screens. One Help screen, for instance, is made for nearly all the Action fields which describes the general properties of these Boolean fields.

Universal Features

There are many features of this menu system that could be applied to others:

- Macros allow modularization of code. Often the code needs to be repeated in a large application. During development, the code can be easily tested from outside the application, e.g. from Display Manager. Version 5 of SAS/AF is very slow and clumsy for testing code itself.

- The ### macros allow generation of macrovariables that can be associated with fields in many program screens. They provide an informational bridge between program screens. In addition fields that have an associated macrovariable will be pre-filled on subsequent use, due to the "memory" of macrovariables. Therefore, it will not be necessary to repeat many keystrokes.

Limitations

The SAS/AF product has certain limitations which make designing an application of this kind a bit trying:

- The === option for writing to an external file. The DDname or fileref associated with this must be hard coded. It would be better if a macro variable or field name could be substituted.

- The program screen for the long job uses two panels. The &_DKEYDEF predefined SAS macrovariable is used to place the cursor in the first field of the second panel. This kind of cursor control limits program screens to only two panels because there are only two directions, FORWARD and BACKWARD.
Pipe Dreams

After designing one large application, the SAS/AF programmer fantasizes about new features SAS/AF could have to make designing future applications easier:

- Automatic page numbering for screens that take up more than one page. As the Command ===> line has been built in, it would be nice if there were a function to put a message such as "Page 1 of 3" on the top right of screens. Customizing and "beautifying" the screens can be very time consuming during development - almost as much as getting everything to work!

- The ability to use macrovariables and pass information to Menu and Help screens.

- Pre-programmed field validation. Like the attribute screen of a program screen allows the association of a macrovariable with a field, it would be very useful for another field to be associated with a field. There are many instances where if one field has a value in it another associated field should have a value (or be blank, as the case may be). The programmer can do consistency checking via the ### macro, but this is a time consuming process. Also, with the latter method, the user can still leave and submit the screen with errors. A 'hard coded' method, such as the field validation via a LIST or LIST screen, would not allow the user to leave with errors.

Requesting the Source Code

For a copy of the source for the macros or the menu system, contact Mr. Vye, at:

Academic Computer Center
University of Rhode Island
Kingston, RI 02881
Phone (401) 792-2501
Fax (401) 789-4040
EAE102@URIACC.BITNET

Acknowledgements

The Technical support staff at SAS Institute were very helpful with queries and patient with bellyaching.

Dave Clayton of the Academic Computer Center at U.R.I. was instrumental in creating the 'DSRESET-loaded' CLIST.

SAS, SAS/AF, and SAS/ETS are registered trademarks of SAS Institute Inc., Cary, NC, USA.

COMPUSTAT is a registered trademark of Standard & Poor's Compustat Services, Inc.

IBM is a registered trademark of International Business Machines Corporation, Armonk, NY.

References


Figure 1.