ABSTRACT

System resources and software are just a few of the factors that impact decisions regarding development of new applications. You should take a closer look at the specifics of SAS/AF software in order to anticipate and plan for both needs and problems that may arise in a new system.

Hardware Considerations

You should consider every detail, even the hardware that will be used. How much does SAS/AF software know about the terminals that are used? The configuration of the screen is stored with the screen, meaning that a screen created on a 30 X 132 configuration could not currently be displayed on a 24 x 80 configuration. Also, if you want to take advantage of colors and other highlighting attributes, the terminal and customization of your controller are important. If your terminal is capable of using the extended highlighting attributes but is not using them you may want to check Item Number 160 in the customization of your control unit.

Fundamentals

Once you have decided on the type of hardware to use, some fundamental knowledge will help you get started. Often-asked questions concern the use of user fields and associated macro variables. A user field is the fill-in-the-blank portion of a PROGRAM screen. The length of the user field is determined by the number + the number of characters in the name of the field + any underscores. SAS/AF software assigns the length based on this information, and it cannot be overridden. A user field, by default, is not a SAS data set variable, nor is it a macro variable. The value of the user field, however, can be assigned to a data set variable or to a macro variable. An associated macro variable is most commonly used to assign the user field value to a macro variable. The name of the associated macro variable is designated on the attribute screen.

Basic Maintenance

The RENAME, DELETE and COPY commands are useful in providing basic maintenance on catalogs and in documenting existing applications. You can use DELETE and COPY commands on the command line of the catalog directory screen in PROC BUILD or on the command line of the catalog directory screen available through full-screen PROC DATASETS. To invoke the RENAME command from the catalog directory screen in PROC BUILD or in full-screen PROC DATASETS, place an 'R' in the unprotected area to the left of the name on the catalog directory screen and then press ENTER. If you issue the COPY command from the command line of the catalog directory screen, the attributes associated with the screen are copied. If you issue the COPY command from the command line of a newly edited screen, attributes are copied if the screen being copied is not in the current catalog. The MERGE = option in PROC BUILD is another method of copying screens. When you are creating a new catalog, the PROC BUILD statement may be followed by a SELECT statement in order to copy only a few desired entries into the new catalog.

The MERGE = option in PROC BUILD is also a way of compressing space in a catalog. With the release of version 5.16, SAS/AF software will also issue an 'Insufficient Disk Space' error message when such a problem exists. This message will be received if the builder attempts to exit the screen by using the END command. If you receive this message, you must issue the CANCEL command in order to exit the screen.

Using the AUTOEXEC Command

Beyond the basics, there are other steps that you can take to insure that the available resources are used to their best advantage. If you do not want to submit commands for your users, or if your users are familiar with the SAS interactive environment, you may decide not to use the SAS Display Manager System. This can save you approximately 80K of memory. However, not using display manager would mean that you could not use the AUTOEXEC feature that allows you to issue DMS commands for your users. The AUTOEXEC command permits users to enter an application without needing to know SAS commands. It can also be used to reset DMS function keys. Note this example in which the DMS function key 6 for the PROGRAM, LOG AND OUTPUT windows is reset:

```
PDS;
QUIT; 
QUIT; 
QUIT;
```

Macros and Macro Variables

There are several facilities available that can enhance your application and make it more efficient. One of these is the use of associated macro variables. Not only do they allow you to pass values from screen to screen, but they can also initialize user field values and permit better user field validation.

Along this same line is the use of macros to construct SAS statements. If you have a segment of code that will be used many times, you may want to put that code in a macro in an AUTOCALL library. Once the macro code is compiled, it will not be recompiled. To invoke these macros, simply type %MACROname or ###MACROname on the program screen. Macros are also useful for conditional execution of PROC and DATA steps. Some special macro variables are available in the SAS/AF software that make user field validation easier. Allow the builder to send customized messages to the message line, prevent the user from exiting the screen until certain requirements are met, verify that subsequent screens exist, and so forth. These automatic macro variables available through the environment of the ###MACRO include the following:

- _DCALL
- _DMSC
- _DERRON
- _DKEY
- _DKEYDEF
- _DALAR
- _DLASTC
- _DCURSOR
- _DCMND

They are documented in Technical Report P-149, 'Changes and Enhancements to the Version 5 SAS System'. When an AUTOCALL macro is used, the option AUTOSOURCE should be specified.
Conditional Execution

The capacity for conditional execution adds flexibility to any application that you implement. Again, special conditional field indicators exist to facilitate this feature in SAS/AF software. The following is a list and a brief description of each:

- `#FIELDNAME`: Submit what follows if FIELDNAME is not blank and a prior `##FIELDNAME` or `~#FIELDNAME` has not suppressed code submission.
- `##FIELDNAME`: Submit what follows if FIELDNAME is not blank. This ends the effect of `#FIELDNAME`.
- `~#FIELDNAME`: Submit what follows if FIELDNAME is blank and a prior `##FIELDNAME` or `~#FIELDNAME` has not suppressed code submission.
- `###FIELDNAME`: Submit what follows if FIELDNAME is blank.
- `##`: Submit what follows unconditionally.

Another feature available is "fast branching." This is achieved through the use of `>>`, `>>>`, or `PPP` indicators. The `>>` indicator causes a branch if the corresponding field is not blank. The `>>>` indicator does an unconditional branch as does the `PPP` indicator. The difference between these two indicators is that the `>>>` branch is scanned for user field substitution and may be suppressed by other conditional execution indicators. Also the `>>>` branch is executed after any `==` processing. This means that you can write to an external file and then fast branch to another screen without reinvoking PROC DISPLAY. Usage of these indicators is highly recommended, when possible, because they can speed up the display of your next screen by as much as 30 percent.

Why is this possible? SAS/AF software is designed with the concept that it will be a front-end to other applications. This means that this software must be able to relinquish control of the program to any DATA or PROC steps that exist in the code. Intuitively, the fewer PROC statements that appear, the less memory that is used and the faster the execution is because the DISPLAY procedure is not being continually reinvoked.

Examples

Given the preceding general glimpse of SAS/AF software and some of its concepts, a more specific look at what occurs in certain situations, and why, is requisite to the creation of more efficient and effective applications. Suppose that a PROGRAM screen has only standard DATA and PROC steps possibly with an associated LIST screen or a list of valid values specified on the attribute screen for a particular user field. When the END key or command is executed that list is checked, and either the user exits the screen, or if errors exist, an error message is displayed on the message line, and the user must type in another value. This scenario becomes more complex as conditional execution, and possibly, the `### MACRO` is used. When is the `### MACRO` executed? It is executed just prior to the PROGRAM screen being displayed, and then again any time the ENTER key or any function key is pressed. Consider the following example where there is one user field called NAME on the screen with an associated macro variable of the same name. The options MACRO and DQUOTE are also set.

How many times would the `%PUT` statement be written to the log? Presuming the screen is displayed to the user and the user types in his name and presses END, the `%PUT` statement would appear on the log in the following form:

```
proc display correct.af.test.program;
  "NAME IS ";
  "NAME IS ANNEU";
run;
```

The first time there is no value for name in the PUT statement because the macro executed before the screen was displayed and the macro variable had not been given a value. The second PUT statement has a value for NAME because the user entered a value and then pressed END.

Expanding upon this idea, review the following example where SCRMME, FIELD2 and FIELD3 all have an associated macro variable by the same name and FIELD2 and FIELD3 additionally have protect and non-display attributes.
In this example, the `### MACRO` would be invoked before the screen is displayed, causing the values of the fields, SCRNME, FIELD2, and FIELD3 to be reset. When the END command is executed, the macro is reinvoked to see if there is a value in the user field, SCRNME. If this field is not null or blank, an unconditional branch is executed to the screen, SUGI1.PROGRAM. If this field is blank, an unconditional branch is executed to the SUGI2.PROGRAM screen. It is important to note that the screen names to which the user will branch do not need to be hardcoded. The following is a portion of the previous code:

```
* Go to SUGI1.PROGRAM if FIELD2 not blank */
&FIELD2 /* Go to screen specified if FIELD2 not blank */
&FIELD2 /* Go to SUGI2.PROGRAM */
```

When you view the log, the efficiency of the `fast branch` is reinforced.

A continuation of the log illustrates the resource usage if two PROC DISPLAY statements are used:

```
/* PROC DISPLAY in.testcat.test.progrUl */
&FIELD2 /* Go to SUGI2.PROGRAM */
&FIELD2 /* Go to screen specified if FIELD2 not blank */
```

You need to make sure that the value typed into the user field, SCRNME, is a valid, existing screen. One method of ensuring that a screen exists is to combine the use of PROC CATOUT and PROC TRANSPOSE with field attributes that would be specified on the attribute screen. Let us take a closer look at one method of accomplishing this type of validation. Assuming the catalog is already created and you need to verify that the entry exists before executing a branch or PROC DISPLAY to that entry, you could execute the following steps:

```
PROC CATOUT CAT=CATOUT; CAT=CATOUT; CAT=
PROC TRANSPOSE CAT=TRANS; CAT=TRANS; CAT=
PROC DISPLAY CAT=TRANS; CAT=TRANS; CAT=
```

PROC CATOUT will create either a data set (if the OUT = option is used) or an external file (if the OUTFILE = option is used) that contains SAS catalog information. One of the pieces of information produced is OBJNAME, which is a character variable whose value is the screen name. PROC TRANSPOSE is used to convert the variable value to a variable. Reviewing the attribute screen, we see that the user field, screen, has the following attributes associated with it:

```
OBJNAME = OBJNAME = OBJNAME =
```
If you want to alter the default key settings, SASUSER may be allocated for the user or the keys may be stored in the catalog in a member called DISPLAY.KEYS. To create a DISPLAY.KEYS member in the current catalog, use PROC BUILD to access the catalog directory screen, and then issue the EDIT command to edit DISPLAY.KEYS. Your must use DISPLAY.KEYS because it is the reserved name for this entry. Edit any keys you want to edit and then use the END command to exit the catalog directory screen. In the search sequence, the catalog would be first in the search order (followed by SASUSER.PROFILE, then WORK.PROFILE, and finally the SASUTL library). However, there are ways to override some commands by using automatic macro variables available in SAS/AF software. The &DKEY and &DCMD macro variables are especially helpful for overriding current key settings:

```sas
/*PRGRM
   Text Editor

   KEY
   
   /KEY KEY
   /*KEY
   /*KEY
   */KEY
*/ FIELD: KEY

/*PRGRM
   Text Editor

   KEY
   
   /KEY KEY
   /*KEY
   /*KEY
   */KEY
*/ FIELD: KEY
```

The preceding example would allow you to create a program screen with the same capability that a menu screen has, in that function keys could be used to execute the desired option.

The previous examples illustrate, on a more basic level, some of the capabilities of SAS/AF software. The constant considerations of hardware, end-user knowledge, flexibility requirements, and environment constraints can be dealt with more positively and efficiently with the implementation of various features and a knowledge of SAS/AF software.

SAS and SAS/AF are registered trademarks of SAS Institute, Inc., Cary, N.C. USA.