CHANGES AND ENHANCEMENTS TO SAS/FSP® SOFTWARE

Patti McAlister
SAS Institute Inc.

ABSTRACT

This paper discusses the changes and enhancements to the SAS/FSP software. Some changes are common to all SAS/FSP procedures, while others affect only a single procedure or a particular screen within a procedure.

All the screens you create with the FSCALC and FSEDIT procedures, letters and forms created with PROC FSLETTER, and function key definition screens are stored in catalogs. Catalogs are not SAS data sets, which can be accessed in the same manner as data sets are currently accessed. Catalog management involves understanding how the catalogs are set up and manipulated.

A common text editor is now available for FSCALC program screens, PROC FSEDIT screen modification, and FSLETTER forms and letters. In addition, PROC FSEDIT has changed significantly in order to protect screens and to regularly store data entered without having to exit the procedure. PROC FSCALC has changed considerably to make it more user friendly and to enable programmers to set up spreadsheets for novice users.

CATALOGS

Catalogs are the key to understanding Version 5 full-screen procedures. A catalog is a member within a SAS library that contains anything produced by a full-screen procedure, such as PROC FSCALC screens or PROC FSLETTER letters or forms. A single catalog can contain entries produced by several procedures. For instance, you may have a data set that is displayed using an FSEDIT screen. You use a catalog to contain the screen. In addition to using the FSEDIT screen with this data set, you may want to create an FSCALC spreadsheet to use with the data set. You can store the spreadsheet in the same catalog as the FSEDIT screen so that it is easier to remember where all full-screen applications are stored for a single data set. You can also give the catalog the same two-level name given to the data set, using the same name helps you remember which catalog belongs to which data set.

Each entry within the catalog also has a two level name. The catalog acts like a mini-library, and entries are differentiated by the name of the entry and also by the entry type. The name of the entry can be any name, but the type is assigned by the procedure that created it and cannot be changed.

Within each catalog you have the capability of deleting, editing, copying, or renaming entries within the catalog. To delete an entry, enter the command

```
DEL name.type
```

on the command line and press the ENTER key. The entry is then removed from the catalog. To edit an entry, you can enter the command

```
EDIT entry.type
```

or you can put an S in the selection field for that entry. The EDIT command only works on entries that are acceptable to the procedure you are using at the time. For example, if you are accessing PROC FSLETTER and try to use an FSCALC report screen, you will be told that it is an invalid type field for this procedure. If you try to edit a letter, however, you can gain access with no problems.

To copy an entry, you need only issue the command

```
COPY name.type
```

An exact duplicate is created in the catalog and should be renamed as soon as possible. If you want to copy an entry from another catalog, you need to use a four-level name. The first two levels specify the libref and catalog. The last two levels specify the name and type. For example, you would issue the command

```
COPY libref.catalog.entry.type
```

where the libref has been defined prior to issuing the command. Once the entry has been created you may want to rename it. To rename an entry, place the cursor on the selection field, type in R, and press the ENTER key. If you have a color terminal, note that the name and the description have changed to yellow. You may now make your changes. When you have finished, press the ENTER key and the changes are stored in the catalog.

Under Version 5 of the SAS System, data sets that are accessed by full-screen procedures and entries in catalogs remain closed until the data set or entry is saved. This method is used to

prevent losing data sets and catalog entries. In addition, screens are now stored preformatted in order to speed up access time. This quicker access time is particularly noticeable with large screens such as PROC FSEDIT screens.

PROFILE CATALOGS

Another concept that is basic to understanding Version 5 full-screen procedures is the profile library. In a profile library you store function key settings for full-screen procedures, including display manager and PROC DATASETS.

When the SAS system is sent to your site, it includes a profile catalog containing all of the default function key settings for each full-screen procedure. In addition, the default forms for PROC FSLETTER are also stored in this catalog. Each site has the capabilty to access this default catalog and change the defaults to those desired by users at the site. Any changes made in this catalog affect the defaults for the entire SAS System at that site.

Some users may wish to set individual defaults that are different from those set in the default system. Each user has the capability to store his own defaults by allocating a library with the reserved libref SASUSER before he enters the SAS System. Any changes made to the function keys while in any full-screen procedure are stored in the catalog SASUSER.PROFILE, a type of SAS file. The easiest way to change these settings is to enter a full-screen procedure and issue the KEYS command. This command displays the default settings, those sent with the SAS System or those that have been modified by the site in the default catalog, which you can then modify. Any modified screen is stored when you exit the function key screen.

Any catalog entry can be stored in SASUSER.PROFILE by accessing it using the appropriate procedure name. For instance, if you want to store a letter called MEMO.LETTER that you use as the basis for all memos in SASUSER.PROFILE, issue the commands

```
PROC FSLETTER LETTER=SASUSER.PROFILE;
RUN;
```

When the catalog directory screen is displayed, you can use the COPY command to copy the letter into SASUSER.PROFILE, if it already exists elsewhere, or you can create the letter while within the procedure.

Whenever a key or form is referenced, a search sequence takes place. The search for the function key setting or form takes place in the current catalog first; that is, the catalog you are currently using. If no match is found for the key setting or form, the system searches to see if the SASUSER libref has been defined. If it has been defined, the system searches the PROFILE entry to see if a match can be found. If the libref has not been defined, a WORK.PROFILE is created for you when you first enter a full-screen procedure, and changes to function keys are stored there for that session. When no SASUSER.PROFILE is found, the system searches the WORK.PROFILE to see if there is a match for the key or form. If no match is found in either SASUSER.PROFILE or WORK.PROFILE, the system then searches the default catalog set by the site for the setting to use. If at any point along the search the system finds a match it stops the search and uses the keys or form found in that library.

ACROSS ALL FULL-SCREEN PROCEDURES

There are several facilities that are common to all the full-screen procedures. The first of these is the interactive HELP facility. Although the HELP facility has existed in the past, it has been changed to make it more user friendly. In addition to extended help menus and information, the command has also been made context sensitive. The last command entered on the command line is the command for which you receive help information. For example, if you have just entered the BAC command on the command line and did not understand the action presented, you could issue the HELP command, and help information on the BACKWARD command would be displayed. To receive the help menu it is necessary only to press the ENTER key to clear the last command; then issue the HELP command. To receive help on a specific command before entering the command, you can issue the HELP command followed by the command keyword to receive specific information.

The text editor is available in all full-screen procedures where editing is available. In PROC FSEDIT, you use the text editor to make modifications to your screen. In PROC FSLETTER and PROC FSCALC, you use the text editor to create your printer control language for forms. In PROC FSLETTER, you can also use the text editor, in addition to the regular FSLETTER text editor to edit a letter. In PROC FSCALC, you can use the text editor to develop your program statements. Editing is made easier for you since the text editor is available in all full-screen procedures that allow editing.

PROC FSCON

A new procedure has been created to convert release 82 full-screen procedure data sets to Version 5 catalogs. This procedure is called PROC FSCON. With the FSCON procedure, there are three possible options to be specified. The SCREEN= option is used to specify the name of the new catalog to be created with either PROC FSEDIT or PROC FSCALC screens. The LETTER= option is used to specify the name of the new catalog to be created with PROC FSLETTER letters and forms. The OLD= option is used to indicate the name of the old existing

screen or letter data set. The OLD= option is required. You must also specify either the SCREEN= or LETTER= option.

PROC FSEDIT

The FSEDIT procedure has undergone many changes with this new release of the SAS System. The most visible change is the removal of the primary menu. You are taken directly into edit mode when the command is issued. You may want to go directly to a new observation screen to begin entering data rather than to enter at observation 1. You can now use the ADD option on the PROC FSEDIT statement to do so. Issue the command

```
PROC FSEDIT ADD;
RUN;
```

Another new option for the PROC FSEDIT statement is the NEW= option. You use this option to create a new SAS data set with zero observations. A full-screen fill-in-the-blank screen is displayed when this command is submitted. You can enter the name of the variable, the variable type (character or numeric), the variable's length, a label, and the informat and format. Once the END command is issued, the FSEDIT edit screen is displayed and you can enter observations.

The DDNAME= option is also specified on the PROC FSEDIT statement. You use it to indicate a sequential file, which has been predefined with a fileref, to be used when letters are sent with the FSEDIT procedure. You must specify the LETTER= option to use the DDNAME= option. Each time a letter (or letters) is sent, it is sent to the fileref used with the DDNAME option. This option is useful when sending letters to a file for further processing or for validation before sending them on to the printer.

Within the procedure, several new commands and options exist. The MODIFY or MOD command is probably the most important. This command allows you to enter the PROC FSEDIT SCREEN MODIFICATION MENU. This menu allows you to make screen modifications, along with several other important functions.

There are five options in the menu. The first option displays a detailed explanation of the way PROC FSEDIT screen modification works. It also displays information on how field identification works. This is very useful information for new PROC FSEDIT users.

The second option in the menu displays PROC FSEDIT SCREEN MODIFICATION. Within this mode, screens are treated as contiguous lines that can be moved, inserted, and copied very easily. This allows you to repeat a page as many times as you want. You can also insert lines without fear of losing the lines at the bottom of your screen. All of the flexibility of the text editor is available to you in SCREEN MODIFICATION MODE so that movement of screens is easily accomplished.

Once you have completed the screen modifications, you press the END key and the FIELD IDENTIFICATION screen is displayed. You can identify the locations of the variables while you are in this mode. Before you are allowed to exit, you must either identify the location that has been changed, or you must make the variable unwanted by using the HOME key. If there are any variables in the unwanted list that you want to add to the screen, you can do so by using the FREE command to make them unwanted and then define their location when prompted, or you can use the DEFINE command to define the variable and its location.

The third option in the modification menu displays FIELD IDENTIFICATION MODE without entering SCREEN MODIFICATION MODE first. You cannot edit the screen while in FIELD IDENTIFICATION MODE.

The fourth option in the menu displays the field attribute screens. You can change the MIN, MAX, INITIAL, and so on, fields. This information is stored with the screen within the catalog.

The fifth option displays a screen where you can define parameter modifications that are stored with the screen. Several key parameters are stored in this screen, which can be important in displaying and saving the data set. The MODIFY password allows you to define a password which is stored with the screen. This gives you screen integrity since the password must be given on the MODIFY command before you are allowed into the SCREEN MODIFICATION MENU.

Another key parameter is the AUTOSAVE parameter. It allows you to define how many observations are to be entered into the data set before it is written out to disk. This gives you the ability to determine when a data set is written out. If an immediate save needs to be done while editing the data set, the SAVE command can be issued.

The STRING and NAME parameters are specified in this screen also. You can specify up to 29 variables for the STRING command. These parameters are stored within the screen catalog so that the commands do not have to be reissued each time you edit the data set. If you need to override the STRING or NAME commands, you can do so by entering the STRING or NAME commands while in EDIT mode.

In the edit screen, you have the ability to send a letter for each observation within the data set. You can do this by issuing the command

```
EDIT lettername
```

You must use the LETTER= option when entering PROC FSEDIT to use this command. Once the letter is displayed, issue the XYZZY command to send the letter with all of the observations in the data set. The command causes letters to be sent beginning at your current observation. For example, if you are at observation 1, and issue the XYZZY command, a letter is sent for each observation in the data set. If you are at observation 100, and issue the XYZZY command, a letter is sent for every observation in the data set beginning with observation 100 and going to the end of the data set.

PROC FSLETTER

You will also see several improvements in the FSLETTER procedure. Among them is the ability to use the text editor available in the other full-screen procedures. When a letter is displayed, the FSLETTER editor is invoked. This editor works the way the old editor worked which means PROC FSLETTER commands can be entered from here. To gain access to the text editor, you must enter the EDIT command on the command line. You cannot issue any PROC FSLETTER commands from the command line of this editor. You must issue the END command to get to the PROC FSLETTER editor and issue any FSLETTER commands from there.

The text editor has also been implemented in the forms screen wherever printer control language is specified. This gives you more space for writing the printer control language for each form.

PROC FSLETTER is also useful as a manipulator of PROC FSEDIT screens. You cannot access the catalog containing PROC FSEDIT screens using the PROC FSEDIT procedure unless you use the LETTER= option. You can use the LETTER= option, or the PROC FSLETTER statement to do catalog management when working with PROC FSEDIT screens.

PROC FSCALC

Major improvements have been made to PROC FSCALC for its production release debut. One improvement has been to the amount of memory used. PROC FSCALC now uses 50% less memory for large spreadsheets than in SAS release 82.

The format of the catalog has changed tremendously. Spreadsheets are stored in catalogs as entries with type specified as CALC. Programs are stored with type PGM and reports are stored with type REPORT. You can also store the FSCALC.PARMS screen and the type FORM to be used with PROC FSCALC. This means you now have the ability to invoke multiple spreadsheets within the same PROC FSCALC session. You can also define multiple programs and multiple reports for the same spreadsheet, as well as multiple spreadsheets for the same program or report. The FSCALC.PARMS screen is used to store information on the form to be used with the PRINT key and the type of defaults you want when you insert new columns or for undefined cells.

A new entry type has been introduced within PROC FSCALC catalogs. The new type is the EXE entry. This catalog entry consists of the compiled program and has the same name as the program from which it was compiled. This entry is useful when programs are very large; using a compiled program can save CPU time.

You can define function keys to execute multiple commands. Separate each command with a semicolon. This enables you to execute one command directly after another such as

```
BAC M;
END;
```

to scroll a spreadsheet backward to the beginning before saving it.

You can edit multiple programs and spreadsheets simultaneously by using special commands such as ZOOM, JUMP, SWAP, and so on, for managing multiple screens. For example, you have edited a spreadsheet and have edited a program screen using the split in the middle of the page. You now have the spreadsheet showing at the top of the screen and the program showing at the bottom. You decide you want the program screen to fill the page, so you enter the ZOOM command while in the program screen, and the program screen is ZOOMed so that it fills the page. To return the program screen and spreadsheet to the former size and position on the screen, simply issue the ZOOM command once more.

The FETCH and CONSOLIDATE commands are now more interactive. You will receive a fill-in-the-blank screen that asks you whether you want to add, subtract, or replace the values. The FETCH screen then asks you if you want to insert or merge. If you want to insert, it asks where you want the insert to take place.

The REPEAT command has been modified to enable you to change the cell and data type of a block of cells. You can also specify whether you want the cells changed, or if you want the column headers, or row names changed. This gives you the ability to change the attributes for column names, row names, or cells.

Another new command is the MENU command. This command is useful to the novice user who wants help in moving, copying, or deleting columns or rows. It is also helpful with many other services.

Several changes have also been made with the program statements available in the program screen. It is now possible to allocate a temporary array using the CALL ALLOCATE statement. The statement is issued as follows:

```
CALL ALLOCATE(arrayname,m,n);
```

where arrayname is the name of the temporary array created. M is the dimension of the array and N is the type of array. There are three types of arrays that can be specified:

1. numeric arrays
2. character arrays
3. undefined arrays.

To free an array you would use the following statement:

```
CALL FREE(arrayname);
```

Another key statement is the PAUSE statement, which allows you to place the spreadsheet in PAUSE mode where you can make changes to the spreadsheet before you continue executing the program using the EXEC command. This is extremely useful for programmers who create spreadsheets and programs for users who do not know PROC FSCALC or the SAS System. The PAUSE statement can be used in conjunction with several other statements to keep control of the spreadsheet by using program statements. Whenever the PAUSE statement is encountered, the _COMMAND_ and _MESSAGE_ special variables are executed. The _COMMAND_ variable contains a character text string to be executed on the command line. For example,

```
_COMMAND_= 'BAC M; END';
```

Whenever the PAUSE statement is encountered, this command is executed. The _MESSAGE_ variable allows you to place a message in the message line of the spreadsheet.

Several other special variables have been created. The _KEY_ variable is a character array of dimension 24 that allows the programmer to set the function keys for this session of the spreadsheet. This allows you to define the function keys to be a limited list of functions that is determined by the application of the spreadsheet.

The _LASTC_ variable is a character variable that contains the last command issued in the spreadsheet. Used along with the _LASTKEY_ variable, which is numeric and contains the number of the last function key pressed, it can be used in IF-THEN statements to control continued processing of the spreadsheet.

PROC FSPRINT

Version 5 also introduces the full documentation of the FSPRINT procedure for the first time. The FSPRINT procedure allows you to browse observations in a SAS data set in full-screen mode. The observations are displayed as a table of rows and columns. The rows are the observations in the data set, while the columns are the variable names.

PROC FSLIST

The FSLIST procedure introduces two new commands. The CURSOR command allows you to specify the row and column position for the cursor to be displayed in the screen. When the CURSOR command is executed without the row and column operands, the cursor returns to the command line.

The ECHO command allows you to control your display while executing the FIND command. If you set ECHO ON, the procedure displays the current page number as the FIND command executes. If you also issue the ECHO DATA command, each page of text is displayed while the FIND command executes.