Look Ma, No DATA Step!
An Introduction to SAS/FSP® Software
Louise Weiler, Pacific Gas and Electric Company

ABSTRACT
SAS/FSP (Full Screen Product) procedures provide convenient interactive facilities for data entry and data presentation. SAS/FSP procedures allow system developers and end-users alike to perform many of the same tasks that would otherwise be performed by the DATA step. These tasks include:
- Creating a SAS® data set
- Getting data into a SAS data set
- Editing data
- Producing customized reports

This tutorial will demonstrate the FSEDIT procedure, not including Screen Control Language (SCL). PROC FSBROWSE and PROC FSPRINT will be mentioned briefly. You will learn how to perform DATA step tasks without writing any code. All you will need to do is respond to prompts appearing on your screen.

This tutorial will also discuss the advantages and the occasional disadvantages of using SAS/FSP.

INTRODUCTION
The title of this tutorial refers to the expression "Look Ma, no hands." The expression evokes the image of a child proudly riding a bicycle with no hands. SAS/FSP makes it possible to perform feats that would otherwise need the "hands" of SAS — the DATA step. With little effort, you can produce a dazzling data entry and data presentation application. But just as there is some danger in riding a bicycle without hands, there are safety tips to consider before and during using SAS/FSP.

The FSEDIT, FSFBRROWSE, and FSPRINT procedures are available in all three versions of SAS currently available — Versions 5.18, 6.03 and 6.06. (In Version 6.06 the FSPRINT procedure has been renamed FSVIEW.) The capabilities and features have been greatly enhanced with each release. For this tutorial, the examples will be presented in Version 6.03.

Often, SAS software gives you more than one way to do the same thing. For example, you can invoke SAS/FSP procedures in two ways: from the Program Editor window, or from SAS/ASSIST®. For this tutorial, the procedures will be invoked using the Program Editor window.

When you enter SAS/FSP, you enter a full-screen environment where you use one-word commands to identify the task you want to perform. There are three ways you can execute commands from within SAS/FSP procedures: by typing commands on the command line, by using function keys, or by selecting commands from a command menu. For this tutorial, commands will be typed on the command line.

Once you are comfortable with the tools presented in this tutorial, then you can proceed to learn Screen Control Language (SCL) which will be covered in another tutorial. While SCL requires that you write code, it greatly enhances the capabilities of SAS/FSP procedures, allowing you complete control over your application.

CREATING A SAS DATA SET
When you create a new SAS data set you need to define the variables that will be input. Within the DATA step, you can define variables using some combination of the INPUT, LENGTH, LABEL, FORMAT, INFORMAT, and ATTRIB statements.

Alternatively, you can create a new SAS data set using either PROC FSEDIT or PROC FSPRINT. Suppose you want to create a SAS data set containing a roster of your co-workers. The roster will contain everyone's name, phone extension, electronic mail id, and birthday. (A real roster would not include the year of the employee's birthday; it is included here for illustrative purposes only.) The data set will reside in a directory referenced by the libname SUGI.

From the Program Editor window, submit the following statements:

PROC FSEDIT NEW=SUGI.ROSTER;
RUN;

Because you have specified the NEW= option, this code will invoke the FSEDIT procedure and place you in data set definition mode. The following screen appears.

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Label</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The upper left hand corner of the screen shows which procedure you are in, what mode you are in, and what data set you are using. The line that shows Command ===> is called the command line. The command line is where you will tell the procedure what task you want to do next, after responding to the prompts on the rest of the screen.

This screen allows you to define variables and their attributes by typing them in the appropriate fields.
Enter the variable names, their type (either character or numeric), their length, label and their format. (Notice that leading zeroes will appear in the phone extension.) What's missing from this screen is a column containing the informat. By using the RIGHT command, an informat column will be displayed in the same area as the format column.

You can toggle between the format and informat column until you have finished defining variables. When you've finished defining the variables, use the END command to create the SAS data set.

If you want to create a data set that is identical to or similar to an existing one, you don't have to supply all the variable definitions again. Instead, use the LIKE= option to identify an existing data set.

PROC FSEDIT NEW=SUGI.NEWROST LIKE=SUGI.ROSTER;
RUN;

Now when you enter data definition mode, the variable names and attributes of the data set you specified are automatically displayed. You have the option of making changes before creating the new data set.

SAFETY TIPS:

For small data sets, the FSEDIT procedure is a quick and easy alternative to the DATA step. For larger data sets, however, I would recommend using the DATA step. If you make a mistake entering the variables in PROC FSEDIT or if you leave out a variable, you can't go back and fix it. When you ride a bicycle without hands, you can only go forward. When you ride with hands, however, you can turn the bike around and go back to where you started. Likewise, code from a DATA step can be retrieved and corrected if you have made an error or left something out.

Another reason for not using PROC FSEDIT is that you can make only one data set at a time. The DATA step gives you much more flexibility. And finally, you should consider in general whether you want to have a record of how your data set is created. FSEDIT leaves no trace; the DATA step can be stored.

GETTING DATA INTO A SAS DATA SET

Once the variables in a data set have been defined, then the variables are ready to receive data. In a DATA step, the data can be obtained from an external file, from the jobstream using a CARDS statement, or from assignment statements. In the FSEDIT procedure, the data is input by the user, observation by observation.

After creating the SUGI.ROSTER file above, you will automatically be placed in observation edit mode and you will see a screen that looks as follows.

To add the first observation to the data set, type the ADD command. Then, type in the data over the underscores. Remember that you have specified MMDDYY8. as the informat for BIRTHDAY.

The observation will be added to the data set when you execute a command. To add another observation, just use the ADD command for as many times as you need. When you are finished, use the END command. The END command saves the data set and ends the PROC FSEDIT session.

Note that an incorrect value has been entered for my birthday. In the following section, you will learn what to do about that problem.

EDITING DATA

The method used above for entering data gave you little protection against entering bad data. The procedure would have supplied an error message if you had entered character data in a numeric field; however, you were able to enter a birthday that was a valid date, but was invalid for this application. The FSEDIT procedure gives you the opportu-
To perform simple error checking, thus preventing users from entering bad data. In this section of the tutorial, we will cover how to correct bad data and how to prevent bad data from being entered.

**CORRECTING BAD DATA:**
To correct the bad data, invoke PROC FSEDIT specifying the existing data set with the DATA= option.

```plaintext
PROC FSEDIT DATA=SUGI.ROSTER;
RUN;
```

You will automatically enter observation edit mode. You can display other observations by scrolling forward or backward, by entering an observation number on the command line, or by searching for an observation based on the value of one or more variables. You can add or delete observations as well as correct existing observations.

To correct bad data, all you need to do is type over the bad data with the correct data. (Here I have put in the year in which I would like to have been born!)

```plaintext
FSEDIT SUGI.ROSTER
Command ===>
NAME: WEILER, LOUISE
PHONE: 8956
BANYANID: LAW4
BIRTHDAY: 05/01/66
```

If you make a mistake during correction you can submit the CANCEL command, and all changes made to the current observation will be cancelled. Once you enter any other command, however, the correction is made to the data set. More specifically, the correction is made to a special temporary data set created for your PROC FSEDIT session. Modifications made to the temporary data set are saved to the original data set whenever you submit the SAVE or END commands. The data set is also automatically saved after 25 modifications. You can change the default of 25 by submitting the AUTOSAVE command and specifying a number higher or lower than 25.

**SAFETY TIPS:**
When you make corrections in place, you no longer have access to the previous version of the data set. If you think you will need the previous version, be sure to make a backup of the original data set. The safest approach is to put corrections to data in a transaction data set containing the same variables as the original data set. That way you can trace changes made to the original data set.

**PREVENTING BAD DATA:**
If you want to prevent implausible data values from being entered into the data set, you can build simple error checking into your application without writing any code. You can accomplish this by building a customized data entry screen. If you want to use this customized screen again, then specify a catalog in which you want your screen stored with the SCREEN= option.

```plaintext
PROC FSEDIT DATA=SUGI.ROSTER
SCREEN=SUGI.ROSTSCRN
MOD;
RUN;
```

These statements invoke PROC FSEDIT and put you into screen modification mode. You will see the following modification menu.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Information about screen modification</td>
</tr>
<tr>
<td>2</td>
<td>Screen Modification and Field Identification</td>
</tr>
<tr>
<td>3</td>
<td>Edit Program Statements and Compile</td>
</tr>
<tr>
<td>4</td>
<td>Assign Special Attributes to Fields</td>
</tr>
<tr>
<td>5</td>
<td>Modification of General Parameters</td>
</tr>
</tbody>
</table>

Option 4 allows you to define or change the special attributes of variable fields that are displayed. There are 13 attributes that can be defined (or not). The two that we can define to check for errors in birthday are the MINIMUM and MAXIMUM attributes. Employees are not likely to be under age 17 or over age 70.

When you select Option 4, you have access to the 13 special attribute panels. The panels look just like your display screen. You can then enter the attribute on top of the underscores for each field. The field attribute panels are stored in the following order, and you can move from panel to panel by using the BACKWARD and FORWARD commands, by the PgUp and PgDn commands, or by typing the name of the panel on the command line.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Panel Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIAL</td>
<td>FATTR</td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>FATTR</td>
</tr>
<tr>
<td>MINIMUM</td>
<td>FATTR</td>
</tr>
<tr>
<td>REQUIRED</td>
<td>PROTECT</td>
</tr>
<tr>
<td>CAPS</td>
<td>JUSTIFY</td>
</tr>
<tr>
<td>FCOLOR</td>
<td>NONDISPLAY</td>
</tr>
<tr>
<td>ECOLOR</td>
<td></td>
</tr>
</tbody>
</table>

The first panel you will see is the INITIAL panel.

```plaintext
FSEDIT Attributes
Command ===>
NAME: *
PHONE: 
BANYANID: 
BIRTHDAY: 
```

If you type MAXIMUM on the command line, the MAXIMUM attribute panel will be displayed, and we can enter the maximum valid birthday.
Likewise if we type MINIMUM on the command line of this panel, the MINIMUM attribute panel will be displayed.

When you submit the END command from an attribute panel you return to the modification menu. When you submit the END command from the modification menu, the attributes are saved in the catalog SUGI.ROSTSCRN. The next time you want to enter or modify data, include this catalog in the SCREEN= option of the PROC FSEDIT statement. When you enter a birthday that is out of range, the procedure will give you an error message and prevent the bad data from being saved.

SAFETY TIPS:
The simple error checking just demonstrated could be improved by using SCL to check actual age as of the day the entry is made. SCL also allows you to perform cross validation between two or more variables. Again, you are urged to learn SCL after this tutorial.

PRODUCING CUSTOMIZED REPORTS
The modification mode in FSEDIT allows you to redesign the display of variables any way you choose. You can make your screen look just like your business forms, surveys, or coding sheets. You can put in any comments, instructions, and labels as well as color and highlighting attributes to help the person entering the data and to help anyone looking at the data.

Even a simple application like the one being demonstrated can be improved with screen redesign. From the modification menu, select Option 2 — Screen Modification and Field Identification. You will enter screen customization mode.

In the first phase of screen customization mode, you can type over any area in the display. You can move, delete or insert any lines in the display. The SAS text editor is available for editing. Underscores must be used to designate each variable field where values can be entered and displayed.

The FSEDIT procedure will continue to prompt you to identify fields until it knows each field's location or knows that the field has been omitted from the display. When PROC FSEDIT knows the location of all variable fields you will receive the message

Note: All fields are identified.

Now you can execute the END command to save the screen and return to the modification menu. Executing the END command from the modification menu takes you back to observation edit mode. Now you will see the data in the redesigned display.
This customized screen is now available for you to use while you are at your terminal browsing through your observations. PROC FSEDIT provides several commands for searching for observations. The commands are FIND, FIND@, NAME, LOCATE, LOCATE:, n, SEARCH, and SEARCH@.

The most powerful of the above commands is the n command, where n is an observation number. When you submit an observation number on the command line, you will go directly to that observation without having to read through another observations first. This command is particularly advantageous when you are working with a large data set. The remaining commands let you search for observations based on the values of one or more variables. The main difficulty with using these commands is that they are not as flexible as the IF statement in a DATA step. You cannot include functions in these commands. It is impossible to construct a search containing both "and" and "or" conditions. It is also impossible to submit a command that uses more characters than are available on the command line.

The customized screen is also available for you to print observations. You can print the PROC FSEDIT displays in two ways. You can use the SPRINT command to do a screen print of an observation.

COMMAND ===> SPRINT NOBORDER NOCMD

The NOBORDER NOCMD options remove the border and the command line that you see on your terminal.

You can also print the entire data set by using the PRINTALL option in the procedure statement.

PROC FSEDIT DATA=SUGI.ROSTER
   SCREEN=SUGI.ROSTSCRN
   PRINTALL;
RUN;

SAFETY TIPS:

Give yourself a large block of time to work on a screen redesign. You cannot leave modification mode until all variables have been identified on the screen or designated as unwanted. The larger your data set, the more time you'll need.

The only disadvantage to this tool is that you cannot get reports that are any wider than the width of your screen. When you write a customized report using the DATA step, you can create extra wide reports. It is possible that PROC REPORT in Version 6.06 will be the best alternative to producing customized reports, but since I don't have Version 6.06 yet, I have not been able to evaluate it.

PROC FSBROWSE

The FSBROWSE procedure is the same as the FSEDIT procedure, except that you can not alter the data set.

PROC FSPRINT (PROC FSVIEW In Version 6.06)

The FSPRINT procedure is similar to the FSEDIT procedure, except that the data is presented in row and column format rather than observation by observation. The FSPRINT procedure produces displays that are a lot like PROC PRINT output. The FSPRINT procedure allows you to change the order of the rows and columns interactively. You can also sort your data set interactively. A special feature allows you to cut and paste the display into letters that have been created by PROC FSLETTER.

SUMMARY

SAS/FSP gives you a lot of power to produce customized data entry and data display applications, even without using Screen Control Language. With SCL, however, you will be able to do nearly everything you can possibly think of. You will be doing tricks on bicycles without hands and without feet!

Comments and questions may be sent to:
Louise Weiler
Pacific Gas & Electric Company
Rates Department
77 Beale Room 1091
San Francisco, CA 94106

ACKNOWLEDGMENTS

My thanks go to Danine Cozzens, Martin Chow, Glenn Ironhat and David Works for their help.

NOTE

SAS, SAS/FSP, and SAS/ASSIST are registered trademarks of SAS Institute Inc., Cary, NC, USA.