Enhancing FSEDIT Procedure Applications

Annette Harris, SAS Institute Inc., Cary, NC

ABSTRACT

With Screen Control Language (SCL) of Version 6 of the SAS® System, the power of the FSEDIT procedure has gained a new perspective. This paper covers cross validation of variable values, customization of messages that can be provided to the end-user, double-entry availablity, and the use of CALL DISPLAY for integration with the SAS/AF® software.

INTRODUCTION

The Version 6 of the SAS System release of PROC FSEDIT provides many new features that make data entry and validation much easier. Additional prompts can be provided for users, double data entry and cross validation of values are available, and CALL DISPLAY can be used to interface with the SAS/AF software. Through a series of small examples, this paper illustrates how simply you can write screen control language programs that will perform these functions and facilitate data entry tasks.

Screen Control Language

Screen Control Language (SCL) is introduced in Release 6.03 of the the SAS System version of PROC FSEDIT. It is a programming language that provides statements and functions that allow you to control the flow of your application. It takes input from your FSEDIT screen and allows you to execute based on that input. You can direct when SCL code will execute through the use of reserved labels. These labels are FSEINIT, INIT, MAIN, TERM, and FSETERM.

FSEINIT is executed when the FSEDIT Procedure is invoked prior to display of the FSEDIT screen. This section is used to initialize variables, open data sets, bring in values from macro variables, and so on.

INIT is executed each time an observation is read. It can be used to initialize variables and display messages.

The MAIN section is executed each time a field is modified and the ENTER key pressed, or an observation is added. PROC FSEDIT then validates values entered by the users and, if no errors exist, executes statements within the MAIN LABEL. Existing errors must be corrected before MAIN will execute.

TERM is executad when an observation is exited or the FSEDIT session is ended. Secondary data sets that are being used can be closed in this section.

SCL statements within the FSETERM label are executed when the END command is used to exit the FSEDIT session. In this section, data sets can be closed and macro variables referenced to save values for a subsequent session.

Applications

To begin creating your FSEDIT application, specify the data set you will be using on the PROC FSEDIT statement. If you want to take advantage of the screen control language or save your modified FSEDIT screen, you also specify a SCREEN name on the FSEDIT statement. For example:

```
PROC FSEDIT DATA=MY.EMPDATA SCREEN=MY.EMPSCRN;
```

To modity the screen and create SCL code, type MOD or MODIFY on the command line of your FSEDIT screen. This command accesses the FSEDIT Modification Menu, where option 2 allows you to update your display screen and option 3 permits creation of SCL statements to control execution during your session. Other options are available for assigning special attributes to fields and specifying general information about the session, including default colors for the window, value to be used for autosave, a modify password, and so on.

Assuming that you have modified your FSEDIT screen and now want to enhance your application by providing some feedback to your user, enter option 3 of the FSEDIT Modification Menu to 'Edit Program Statements and Compile'. Customized messages are now available through the use of an SCL system variable, _MSG_, so we begin our SCL example as follows:

```
FSEINIT:
_MSG_ = 'Welcome to the Employee Information System';
return;

INIT:
return;

MAIN:
return;

TERM:
return;

FSETERM:
return;
```

The customized message is nice, but suppose that you would like to be able to bring in some information from another data set to cross- validate data entered on your FSEDIT screen and provide informative messages. You would then need to take advantage of some of the SCL functions that open data sets and locate and return variable names and values.

In the next example, a secondary data set is accessed to provide more information about the employee whose unique code number you enter on your FSEDIT screen. For illustration purposes, our current data set (the one we are viewing with PROC FSEDIT) has the employee name, code, address, and phone number. Another SAS data set contains variables whose values are employee code, years of education, years of tenure with the company, and the department the employee works in. If the employee code typed is invalid, a message is issued and the field is highlighted in error. If a valid code is provided, information about the employee name, address, and phone number are displayed on the screen, along with a message noting the employee name.

A short description of the functions used will help clarify the example. The OPEN function opens the SAS data set; 'U' or 'I' can be specified to indicate the UPDATE or INPUT mode. VARNUM returns the number of the variable in the data set (and can be used to ensure the variable exists in the data set). CALL WDEF is used to resize the window so a legend can be written at the top. MODIFIED indicates whether a field has been updated. The LOCATEC function locates a specific character value in the data set. There is a counterpart LOCATEN function available if the value you wish to search is numeric. An error flag is set for the field with ERRORON. GETVARC fetches the value of a specific character variable from the current observation. It also has a counterpart function, GETVARN, for fetching a numeric variable

value. Many of the functions return values that are useful as arguments to other functions. CALL CLOSE is used to close the data set.

```
FSEINIT:
  DSID = OPEN('A.EMP','U');
  NUM  = VARNUM(DSID,'EMPCODE');
  _MSG_ = 'WELCOME TO THE EMPLOYEE INFORMATION SYSTEM';
  CALL WDEF(7,1,18,80);
 RETURN;

INIT:
RETURN;

MAIN:
  IF MODIFIED(EMPCODE) THEN DO;
     VER = LOCATEC(DSID,NUM,EMPCODE,'U',' ');
       IF VER = 0 THEN DO;
            ERRORON EMPCODE;
            _MSG_ = 'THE CODE ENTERED IS INVALID.  PLEASE RESPECIFY.';
            END;
       ELSE DO;
       ERROROFF EMPCODE;
          NAME = GETVARC(DSID,VARNUM(DSID,'NAME'));
          _MSG_ ='INFORMATION ON' || ' '|| NAME || ' '|| 'IS DISPLAY    ED';
          ADDR = GETVARC(DSID(VARNUM(DSID,'ADDR'));
          PHONE = GETVARC(DSID,VARNUM(DSID,'PHONE'));
       END;
  END;
  RETURN;

TERM:
RETURN;

FSETERM:
  CALL CLOSE(DSID);
RETURN;
```

An additional step can provide a list of employee codes if valid codes are unknown. With a few additional statements, requested when the user types a question mark in the employee code field, PROC FSPRINT is invoked to provide a listing of the variable values in the data set. A legend has been added to notify the user that he can type a question mark to receive a listing of the data set variables and their values.

CALL PUTLEGEND and CALL LEGEND have been added to the example code to create a legend and display it. CALL FSPRINT starts an FSPRINT session that allows the user to view variables and their values in the specified SAS data set.

```
FSEINIT:
  DSID = OPEN('A.EMP','U');
  NUM  = VARNUM(DSID,'EMPCODE');
  _MSG_ = 'WELCOME TO THE EMPLOYEE INFORMATION SYSTEM';
  CALL WDEF(7,1,18,80);
RETURN;

INIT;
CALL PUTLEGEND(1,' ',BLUE', 'REVERSE');
CALL PUTLEGEND(2,' ',BLUE', 'REVERSE');
CALL PUTLEGEND(3,'IF YOU WOULD LIKE TO SEE A LISTING','BLUE','REVERSE   ');
CALL PUTLEGEND(4,'TYPE A ? IN THE EMPLOYEE CODE FIELD,'BLUE','REVERSE   ');
CALL LEGEND();
CALL WDEF(7,1,18,80);
RETURN;

MAIN:
  IF (EMPCODE = '?' ) THEN DO;
    CALL FSPRINT('A.EMP', 'BROWSE');
    END;
  ELSE
  IF MODIFIED(EMPCODE) THEN DO;
     VER = LOCATEC(DSID,NUM,EMPCODE,'U',' ');
       IF VER = 0 THEN DO;
            ERRORON EMPCODE;
            _MSG_ = 'THE CODE ENTERED IS INVALID.  PLEASE RESPECIFY.';
             END;
       ELSE DO;
          ERROROFF EMPCODE;
          NAME = GETVARC(DSID,VARNUM(DSID,'NAME'));
          _MSG_ ='INFORMATION ON' || ' '|| NAME || ' '|| 'IS DISPLAYED';
          ADDR = GETVARC(DSID(VARNUM(DSID,'ADDR'));
          PHONE = GETVARC(DSID,VARNUM(DSID,'PHONE'));
       END;
  END;
  RETURN;

TERM:
RETURN;

FSETERM:
  CALL CLOSE(DSID);
RETURN;
```

Other methods of providing valid data values exist. An alternative method is to integrate use of the SAS/AF product into this application. This allows you greater control over the display of your data set. CALL DISPLAY is used to invoke a SAS/AF program screen that displays a listing. This is accomplished through use of extended tables. Extended tables are a facility for displaying a set of repeated fields on a PROGRAM entry. (Extended tables are available only in SAS/AF software.) Assuming other sections remain the same, the MAIN section is modified as follows:

```
MAIN:
  IF (EMPCODE = '?' ) THEN DO;
    CALL DISPLAY('LIST.PROGRAM');
    CALL PUTLEGEND(1,' ',BLUE', 'REVERSE');
    CALL PUTLEGEND(2,'PLEASE SPECIFY A VALID CODE','BLUE','REVERSE');
    CALL PUTLEGEND(3,' ',BLUE', 'REVERSE');
    CALL PUTLEGEND(4,' ',BLUE','REVERSE');
    CALL LEGEND();
    END;
  ELSE
  IF MODIFIED(EMPCODE) THEN DO;
     VER = LOCATEC(DSID,NUM,EMPCODE,'U',' ');
       IF VER = 0 THEN DO;
            ERRORON EMPCODE;
            _MSG_ = 'THE CODE ENTERED IS INVALID.  PLEASE RESPECIFY.';
            END;
       ELSE DO;
          ERROROFF EMPCODE;
          NAME = GETVARC(DSID,VARNUM(DSID,'NAME'));
          _MSG_ ='INFORMATION ON' || ' '|| NAME || ' '|| 'IS DISPLAYED';
          ADDR = GETVARC(DSID(VARNUM(DSID,'ADDR'));
          PHONE = GETVARC(DSID,VARNUM(DSID,'PHONE'));
       END;
  END;
  RETURN;
```

Because the INIT section would not be executed again upon return to the FSEDIT screen, new legend information is needed, or the window resized if you want the screen to cover the entire display area.

Using a different example, we can illustrate a double-data entry application. Assuming that data have already been entered into an original data set, the following SCL code would be used to validate and provide feedback when data is entered into the counterpart data set. The variable names in the data set are drug (a character variable), amount (a numeric variable) and RESULT (a numeric variable). On the test screen, additional fields, CON1, CON2, and CON3, are provided so data can continue to be entered even if an inconsistency is found. CALL EXECCMD is used to execute the OVERRIDE command. Later a data step can be executed to report any differences in variable values between the two data sets.

```
FSEINIT:
  DSID = OPEN('A.DRUG','U');
  DRUGNUM = VARNUM(DSID,'DRUG');
  AMT = VARNUM(DSID,'AMOUNT');
  RES = VARNUM(DSID,'RESULT');
RETURN;

INIT:
  _MSG_ = 'PLEASE ENTER DATA IN THE APPROPRIATE FIELDS';
RETURN;

MAIN:
```

```
IF MODIFIED(DRUG) THEN DO;
  DRG = LOCATEC(DSID,DRUGNUM,DRUG,'U',' ');
  IF DRG = 0 THEN DO;
    ERRORON DRUG;
    _MSG_ = 'DRUG NAME IS INVALID.  PLEASE VERIFY';
  END;
     IF CON1 = 'Y' THEN DO;
       CALL EXECCMD('OVERRIDE');
     END;
END;
IF MODIFIED(AMOUNT) THEN DO;
 AMT = LOCATEC(DSID,AMT,AMOUNT,'U',' ');
  IF AMT = 0 THEN DO;
    ERRORON AMOUNT;
    _MSG_ = 'VALUE FOR AMOUNT DOES NOT MATCH.  PLEASE VERIFY.';
  END;
     IF CON2 = 'Y' THEN DO;
       CALL EXECCMD('OVERRIDE');
     END;
END;
IF MODIFIED(RESULT) THEN DO;
  TRES = LOCATEC(DSID,RES,RESULT,'U',' ');
  IF TRES = 0 THEN DO;
    ERRORON RESULT;
    _MSG_ = 'VALUE FOR RESULT DOES NOT MATCH.  PLEASE VERIFY.';
  END;
     IF CON3 = 'Y' THEN DO;
       CALL EXECCMD('OVERRIDE');
       REFRESH;
     END;
END;
RETURN;

TERM:
RETURN;

FSETERM:
  CALL CLOSE(DSID);
RETURN;
```

CONCLUSION

These examples illustrate some of the enhancements available to PROC FSEDIT applications through the use of the screen control language. Many other SCL functions exist that provide you the opportunity to deal more positively and efficiently with customization of full-screen applications.