Creating Your First Data Entry System Using PROC FSEDIT

Steven A. Wilson, Resource Biometrics, Inc., Emeryville, CA 94608

Abstract

The SAS/FSP® product contains two useful procedures, PROC FSEDIT and PROC FSBROWSE, which allow SAS System® users to edit and browse a SAS data set in a full-screen, interactive mode. Fully customizable screens may also be built via these procedures for presenting the data. When these screens are properly designed and implemented, they allow you to create a fully functioning data entry system.

While the majority of issues discussed in this paper concern PROC FSEDIT basics, there are other issues that must be addressed to turn a PROC FSEDIT screen into a data entry system. This paper will show you how to develop a simple data entry system using the SAS System. In ten (10) simple steps a basic, stand-alone data entry system will be designed that can be used without any knowledge of the SAS System.

Another important feature available for use with PROC FSEDIT screens is the SAS Screen Control Language (SCL). SCL allows you to control actions to be performed as data is entered into the screen. Only a minimal amount of SCL is discussed in this paper.

Introduction

When using PROC FSEDIT, users may enter new observations into a SAS data set as well as modify existing observations in a SAS data set. This procedure also allows the construction of screens to display the data in any format you desire, giving you the ability to create custom data entry screens.

However, knowledge of the PROC FSEDIT procedure alone is not really enough for you to be able to create a data entry system. There are other aspects that must be considered such as design, maintenance, user acceptance, and ease of use. By following these ten, easy steps, you will be able to create a fully functioning data entry system.

10 Easy Steps

1) Design
2) Create an Empty Data Set
3) Save the Data Set Attributes
4) Start PROC FSEDIT
5) Create FSEDIT Screen
6) Set FSEDIT Keys
7) End PROC FSEDIT
8) Save FSEDIT Code
9) Create User Access
10) Use Your System

1) Design

■ Determine what variables are needed in your data set. Not all variables in a data set need be displayed on the created data entry screen.

■ Mock-up a dummy screen on paper. This helps you design a good data entry screen before starting to build it on the computer. Review this paper copy with the end user and obtain approval.

■ Determine what data sources need to be accessed to obtain data elements to display on the screen.

2) Create an Empty Data Set

■ Write a simple SAS DATA step to create an empty data set. This step is only necessary when you do not have an existing data set for which you are building a screen.

Note that although you can create a new SAS data set within PROC FSEDIT, I prefer this method because source code is generated that can be later re-used.

■ The code below creates an empty data set containing 5 variables. Type this code into your program window and submit it to SAS for execution.

```
LOG
Command ===>

NOTE: AUTOEXEC processing beginning; file is /sas609/autoexec.sas.

NOTE: SAS initialization used:
      real time           0.67 seconds
      cpu time            0.45 seconds

PROGRAM EDITOR
Command ===> SUBMIT
00001 DATA SASUSER.SUGIDEMO ;
00002   ATTRIB  ID       LENGTH = $8   LABEL = 'Record ID' ;
00003   ATTRIB  SEX      LENGTH = $1   LABEL = 'Subject Gender'
00004                                  FORMAT = $SEX. ;
00005   ATTRIB  DATE     LENGTH = 5    LABEL = 'Admission Date'
00006                                  FORMAT = MMDDYY8.   INFORMAT=MMDDYY6.;
00007   ATTRIB  PHONE    LENGTH = $14  LABEL = 'Phone Number' ;
00008   ATTRIB  COMMENT  LENGTH = $200 LABEL = 'Notes' ;
00009   STOP ;
00010 RUN ;
```

Source Code Notes:

- A LIBNAME statement is required to store your data set in a SAS library other than SASUSER.
- The ATTRIB statement allows you to define variables in the created data set.
    - LENGTH defines the number of bytes for storage.
    - FORMAT defines a SAS format used for displaying the variable's value on the screen.
    - INFORMAT defines a SAS informat used for reading data as it is entered into the screen.
- The STOP statement causes the data set to have zero (0) observations.

■ Alternative code could be written if you want to pull data from elsewhere, merge the data with other data sources, or perform other types of pre-processing on the data prior to making it available to be edited. Care should be taken in these situations to prevent someone else from updating the data while the data entry system is in use.

3) Save Data Set Attributes

■ It is important to save the created DATA step in case you realize, during testing, that your data set is inadequate.

■ The easiest way to do this is to RECALL the previously submitted code and FILE it to an external file (file in a DOS directory).

```
LOG
Command ===>

60     STOP ;
61   RUN ;

NOTE: Variable ID is uninitialized.
NOTE: Variable SEX is uninitialized.
NOTE: Variable DATE is uninitialized.
NOTE: Variable PHONE is uninitialized.
NOTE: Variable COMMENT is uninitialized.
NOTE: The data set SASUSER.SUGIDEMO has 0 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.76 seconds
      cpu time            0.24 seconds
```

```
PROGRAM EDITOR
Command ===> RECALL
NOTE: 10 Lines submitted.
00001
00002
```

```
LOG
Command ===>

NOTE: The data set SASUSER.SUGIDEMO has 0 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.64 seconds
      cpu time            0.27 seconds
```

```
PROGRAM EDITOR
Command ===> FILE 'A:\SUGIDEMO\CRE8DATA.SAS'
NOTE: 10 Line(s) recalled.
00001 DATA SASUSER.SUGIDEMO ;
00002   ATTRIB  ID       LENGTH = $8   LABEL = 'Record ID' ;
00003   ATTRIB  SEX      LENGTH = $1   LABEL = 'Subject Gender'
00004                                  FORMAT = $SEX. ;
00005   ATTRIB  DATE     LENGTH = 5    LABEL = 'Admission Date'
00006                                  FORMAT = MMDDYY8.   INFORMAT=MMDDYY6.;
00007   ATTRIB  PHONE    LENGTH = $14  LABEL = 'Phone Number' ;
00008   ATTRIB  COMMENT  LENGTH = $200 LABEL = 'Notes' ;
00009   STOP ;
00010 RUN ;
```

■ If the created data set needs modified during development or testing, you can recall this saved code, modify it and then re-create an empty data set.

■ To keep any existing data in the data set when modifying it, simply change the STOP statement to a SET statement that reads in the data set. It is *important* that the ATTRIB statements come before the SET statement so any changes to the variables' attributes will be defined in the newly created data set.

■ After filing the DATA step code, CLEAR the Program Editor window.

4) Initiate PROC FSEDIT

■ PROC FSEDIT is the SAS System procedure for creating user friendly data entry screens.

■ You must specify both the existing SAS data set and the FSEDIT screen that is to be used. If the specified screen does not exist, a default screen will be displayed that shows all variables in the data set.

■ The LABEL option of the PROC FSEDIT statement is often useful when creating the default screen. This option causes PROC FSEDIT to use the variable labels instead of the variable names when creating a default screen.

■ A screen name has four (4) parts:

SCREEN = *libref.catalog.entry.entry-type*

The catalog *entry-type* (the 4th level of the name) must be SCREEN

```
LOG
Command ===>

NOTE: Variable SEX is uninitialized.
NOTE: Variable DATE is uninitialized.
NOTE: Variable PHONE is uninitialized.
NOTE: Variable COMMENT is uninitialized.
NOTE: The data set SASUSER.SUGIDEMO has 0 observations and 5 variables.
NOTE: DATA statement used:
      real time             0.76 seconds
      cpu time              0.24 seconds
```

```
PROGRAM EDITOR
Command ===> SUBMIT
NOTE: Lines have been cleared.
00001
00002 PROC FSEDIT  DATA   = SASUSER.SUGIDEMO    LABEL
00003              SCREEN = SASUSER.SUGIDEMO.SCREEN.SCREEN ;
00004 RUN ;
00005
00006
00007
```

Screen Naming Tips:
Keep the created screen in the same SAS library as the data set it is used with. This is illustrated in the above example.
eg.

```
SCREEN =
SASUSER.SUGIDEMO.SCREEN.SCREEN
```

However, if the screen is being developed as part of a multi-screen system, you should keep all screens in one catalog named SCREENS with each separate screen named for the data set the screen is used to view.
eg.

```
SCREEN =
SASUSER.SCREENS.SUGIDEMO.SCREEN
```

5) Create the Screen

■ If the specified screen does not exist, a default screen is created and displayed. Usually you will want to enhance this default screen.

■ Notice that the default screen has default data entry fields, denoted by the underscores (_) on the screen. These fields are of an appropriate length to display the formatted value of a variable.

■ From the default PROC FSEDIT screen, issue the MODIFY (or MOD) command

```
FSEDIT SASUSER.SUGIDEMO                                          Obs 0
Command ===> MOD
WARNING: No observations on data set.  Please enter END or ADD.
 Record ID      _____                        Subject Gender  ____

 Admission Date _____

 Phone Number   __________

 Notes          ____________________________________________________
 ____________________________________________________________________
                                                                    R
```

■ The FSEDIT screen modification menu appears below.

```
FSEDIT Menu
Select Option ===>

          1 Information about screen modification
          2 Screen Modification and Field Identification
          3 Edit Program Statements and Compile
          4 Assign Special Attributes to Fields
          5 Modification of General Parameters
          6 Browse Program Statements
                                                                    R
```

■ The screen modification menu options perform the following actions:
1) Help for screen modification
2) Design screen layout and identify variables to be displayed
3) Enter and compile Screen Control Language (SCL) code

4) Field attributes are attributes that can be assigned to each variable displayed on the screen. These attributes include minimum and maximum values, initial values, and can indicate which variables are required or protected.
5) General paremeters are attributes that control overall screen settings.
6) Browse any entered SCL code.

Each menu item on this FSEDIT Modification Menu is discussed in the following sub-steps.

5) Create the Screen - Screen Modification

■ Select Item 2 (Screen Modification and Field Identification).

■ The data entry screen is displayed in the SAS text editor. You have complete freedom to enter any text on the screen.

■ Turn NUMS on to allow the use of various line editting commands, if desired.

```
FSEDIT Modify
Command ===> NUMS

Record ID      _____                    Subject Gender  ____

Admission Date _____

Phone Number   __________

Notes          ____________________________________________*
               ____________________________________________*
               ____________________________________________
                                                          R
```

```
FSEDIT Modify
Command ===>
NOTE: NUMS is set to ON.
00001
00002 Record ID      _____                    Subject Gender  ____
00003
00004
00005
00006 Admission Date _____
00007
00008
00009
00010 Phone Number   __________
00011
00012
00013
00014 Notes          ____________________________________________
00015 ____________________________________________
00016 ____________________________________________
00017
00018
00019
00020                                                     R
```

■ Edit your screen to appear as you desire.

■ Space for variables to be displayed must be represented by using an underscore (_). Use one underscore for each character of the variable value to be displayed.

■ The variable field *must* have a leading and a trailing blank space.

■ A screen may use more space than can be displayed on a physical screen. In otherwords, an FSEDIT logical screen may be comprised of one or more physical screens that the user may scroll through using the LEFT and RIGHT commands.

■ For character variables, the underscores for a field do not have to be contiguous. Use an asterisk (*) at the end of a series of underscores to indicate that the character field continues in the next set of underscores encountered to the right or below the current field. This is also useful for fields that are longer than the physical screen, as seen for the COMMENT variable field on the default screen. For example:

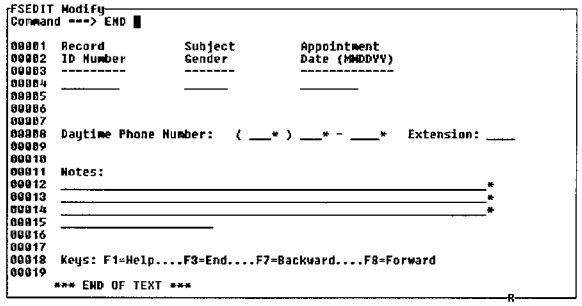
SSN: ____* - __* - _____

■ Important hard coded text to include on your screen:

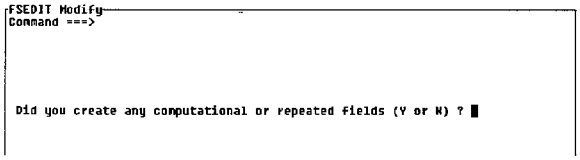
- Instructions for data entry
- Description of function keys
- Appropriate appearance

■ On the PC, text may be enhanced by using the COLOR TEXT PINK command to color the text entered after issuing the command. Other colors are available, but for the sake of clarity, do not use more than three colors on a screen.

■ After the screen is built, END the screen modification session.



■ Answer N to the following prompt, unless, of course, you have created a computed or repeated field on the screen (see SAS/FSP Software documentation).



■ Computed fields are fields that display values that are computed, using Screen Control Language, as data is loaded and/or modified into the screen

■ Repeated fields are used when the same variable is to be displayed on multiple physical screens.

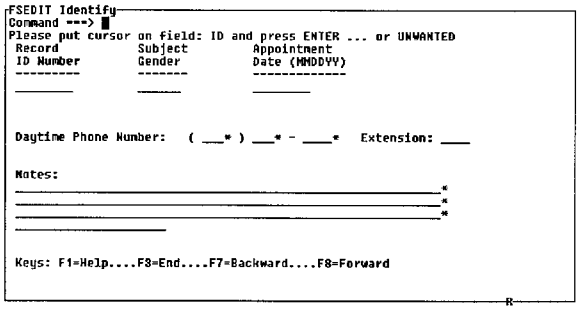
5) Create the Screen - Field Identification

■ Upon exiting the Screen Modification Mode, you will be asked to identify the placement of each variable on the modified screen. This identification is only necessary for those fields that SAS cannot identify. SAS will be unable to identify any fields on a line that has been modified.

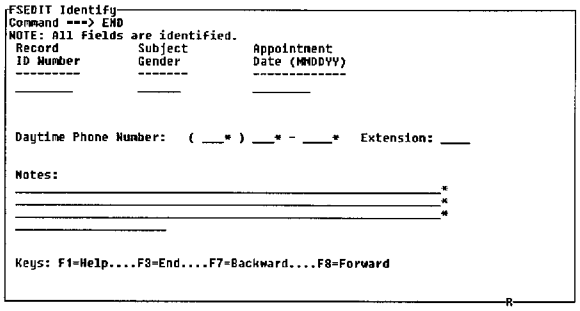
■ Two particularly important commands are available for use while in the Field Identification Mode:

- UNWANTED
  This is a command that can be used when asked to identify a variable that you do not wish to display on the built screen. When used, the variable will remain in your data set but will never be displayed by the screen.
- WANT
  Issue the WANT command to display variables that have been previously identified as unwanted.

■ The Field Identification Mode screen is shown below:

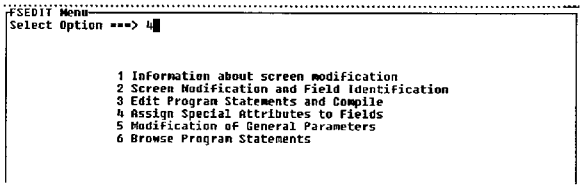


■ When all fields have been identified, END the Field Identification Mode. You will be returned to the FSEDIT screen modification menu.

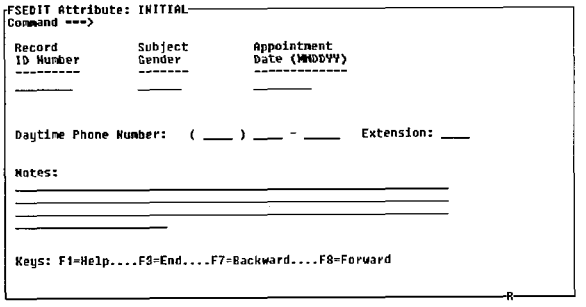


5) Create the Screen - Field Attributes

■ Select option 4 (Assign Special Attributes to Fields). This allows you to customize each field appearing on the screen.



■ The first field attribute is the INITIAL screen. This allows specification of an initial value to be automatically entered when a new observation is added. This INITIAL field attribute screen is shown below.

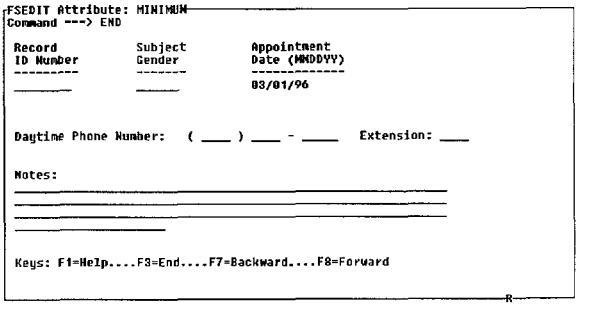


■ Scroll FORWARD to allow specifying other field attributes. Some important attributes are described below:

MIN	Error flagged if entered value is less than the specified minimum.
MAX	Error flagged if entered value is greater than the specified maximum.
REQUIRED	Indicates which fields are required. You cannot save or exit the observation until these fields are entered.
JUSTIFY	Justification of entered data. Left justification is useful for numeric fields, since this allows for easy over-typing of values in these fields.
CAPS	Indicates which fields are automatically converted to upcase.
PROTECT	Indicates which displayed fields are protected from modification. The cursor will skip these fields.
PAD	Specify a pad character for the field. Used to change the underscores that indicate an entry field into another character. The pad character can be set to a blank, creating a "blank" field that will only display when a value is present. This is a useful feature for displaying messages on the screen.

■ When finished entering field attributes, END the field attribute assignment screen to return to the FSEDIT screen modification menu.

In the example below, a minimum value for date has been specified on our sample screen.



5) Create the Screen - Screen Attributes

■ Select option 5 (Modification of General Attributes) to allow the specification of screen attributes.

■ Most important among these attributes are the following:

- Name Command Variable
  Specify the variable that will most often be used to locate an observation in the data set as the Name Command Variable.
- String Command Variables
  Specify variables that will be used for searching through the data set in the String Command Variables list.
- Keys Name
  Specify a function key catalog entry that contains the function keys desired for use with this data entry screen (see Step 6). This catalog entry *must* be contained in the same catalog as the data entry screen and can be used by multiple screens.

```
FSEDIT Parms
Command ===> END

Area          Color      Attribute

Background    BLACK
Border        WHITE      NONE            Allow DELETE  command: Y
Banner        WHITE      NONE            Allow ADD/DUP command: Y
Command       CYAN       NONE
Message       YELLOW     NONE
Protected     CYAN                       Keys name:   SUGIDEMO
Unprotected   YELLOW

Override on errors:       Y              Window rows:      24
Override on required:     Y              Window cols:      80
Autosave value:           25             Start row:         1
Modify password:          _____          Start col:         1
Name command variable:    ID

String command variables: COMMENT _____ _____ _____ _____
_____ _____ _____ _____ _____ _____ _____ _____
_____ _____ _____ _____ _____ _____ _____ _____
                                                          R
```

■ Other options include specifying the AUTOSAVE value, disabling the ADD, DUPLICATE, DELETE, and/or OVERRIDE commands, and specifying screen size.

■ Screen colors can also be specified here, but some host systems do not work well with these specifications.

■ When finished with the general attributes, END back to the Screen Modification menu.

5) Create the Screen - SCL Program

■ You may optionally select option 3 (Edit Program Statements and Compile) to enter Screen Control Language (SCL) for custom data entry actions.

■ SCL is an extremely powerful SAS-like language that allows you to control actions to be performed as data is entered into a screen. Almost any kind of processing imaginable can be performed using SCL, but I recommend starting off slowly with SCL - the learning curve can be substantial.

■ Use NUMS command to get line numbers on the FSEDIT Program screen.

■ All SCL program must have three (3) labeled sections:

INIT: Actions performed as new observation brough into the screen

MAIN: Actions performed each time an observation is modified

TERM: Actions performed as an observation leaves the screen

Additional labeled sections are available in an FSEDIT screen:

FSEINIT: Actions performed when FSEDIT is started

FSETERM: Actions performed when ending FSEDIT

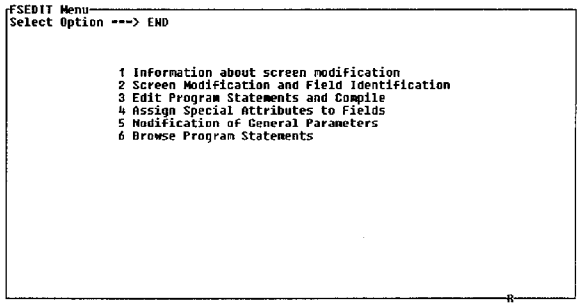
■ Only the most elementary usage of SCL is shown below. This shows how to code SCL to perform data entry verification.

```
FSEDIT Program
Command ===> END
NOTE: NUMS is set to ON.
00001 INIT :
00002 RETURN ;
00003
00004 MAIN :
00005   ERROROFF _ALL_ ;
00006   IF DATE GT TODAY() THEN DO ;
00007     ERROROW DATE ;
00008     _MSG_ = 'Invalid Date';
00009     RETURN ;
00010   END ;
00011   ELSE IF SEX EQ PUT(SEX,$SEX.) THEN DO ;
00012     ERROROW SEX ;
00013     _MSG_ = 'Invalid Sex Code' ;
00014     RETURN ;
00015   END ;
00016 RETURN ;
00017
00018 TERM :
00019 RETURN ;
00020
                                                          R
```

■ It is preferrable to perform as much edit checking as possible by using informats or defining field attributes (MIN, MAX, etc.). Since a minimum value was specified as a field attribute for DATE, there is no need to verify that the DATE is greater than the minimum acceptable date.

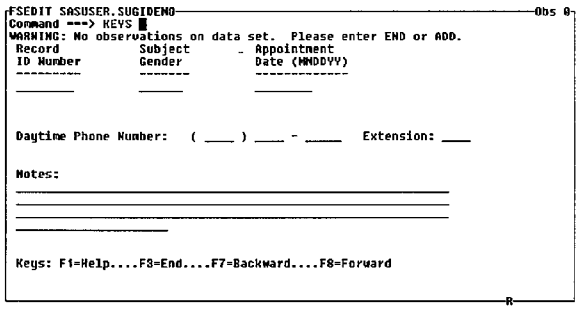
■ After entering your SCL program, END the FSEDIT Program screen. The SCL code will be compiled and checked for errors. You will be notified if any problems are encountered.

■ END the FSEDIT screen modification menu. Your new screen will now be displayed. Your screen is now complete, and this is the end of Step 5, Creating the FSEDIT Screen.



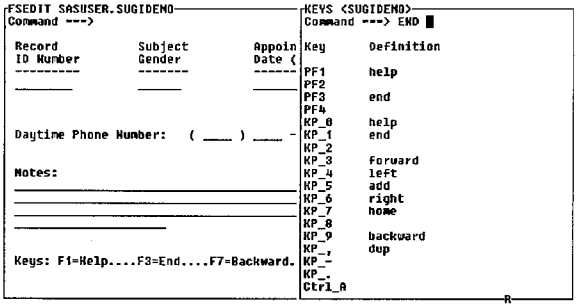
6) Set FSEDIT Keys

■ Issue the KEYS command to view the function key settings for your screen.



■ If no keys entry was made in the Screen Attributes section, then the screen will use the user's default FSEDIT keys. This is usually not desirable since these keys will be used in all default FSEDIT sessions.

■ When a new keys entry has been defined in the Screen Attributes, a blank keys window will appear. It is recommended that you set these function keys to perform useful functions while using the created screen. My newly assigned function FSEDIT keys are shown below.



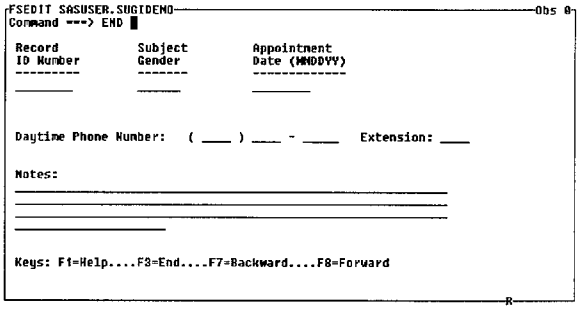
■ These function key definitions will only be used by this screen (or any other screen that have these keys assigned). This is much cleaner because you can customize the keys to match the screen rather than relying on the user's default FSEDIT key settings.

■ Important commands should have function keys. At a minimum, these commands should be assigned to function keys in an FSEDIT screen:

HELP	END
FORWARD	BACKWARD
ADD	DUP
LEFT	RIGHT

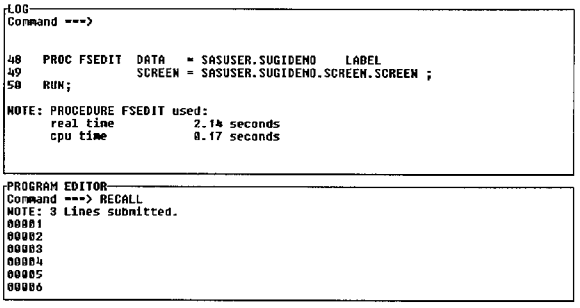
7) End FSEDIT

■ Now that your screen is built and the FSEDIT keys have been set, you can exit the FSEDIT procedure. Simply issue the END command (or press the appropriate function key). This ends the FSEDIT procedure and returns you to the SAS Display Manager.



8) Save FSEDIT Code

■ As in Step 3, RECALL the previously submitted code and FILE it to an external file.



```
LOG
Command --->

48   PROC FSEDIT  DATA   = SASUSER.SUGIDEMO    LABEL
49                SCREEN = SASUSER.SUGIDEMO.SCREEN.SCREEN ;
50   RUN;

NOTE: PROCEDURE FSEDIT used:
      real time           2.14 seconds
      cpu time            0.17 seconds
```

```
PROGRAM EDITOR
Command ---> FILE 'A:\SUGIDEMO\FSEDIT.SAS'
NOTE: 3 Line(s) recalled.
00001 PROC FSEDIT  DATA   = SASUSER.SUGIDEMO    LABEL
00002              SCREEN = SASUSER.SUGIDEMO.SCREEN.SCREEN ;
00003 RUN;
00004
00005
00006
```

9) Create User Access - Single User System

■ CLEAR the Program Editor window and issue the KEYS command.

```
LOG
Command --->

48   PROC FSEDIT  DATA   = SASUSER.SUGIDEMO    LABEL
49                SCREEN = SASUSER.SUGIDEMO.SCREEN.SCREEN ;
50   RUN;

NOTE: PROCEDURE FSEDIT used:
      real time           2.14 seconds
      cpu time            0.17 seconds
```

```
PROGRAM EDITOR
Command ---> CLEAR ; KEYS
NOTE: 3 line(s) written to external file.
00001 PROC FSEDIT  DATA   = SASUSER.SUGIDEMO    LABEL
00002              SCREEN = SASUSER.SUGIDEMO.SCREEN.SCREEN ;
00003 RUN;
00004
00005
00006
```

■ Assign a function key as shown below in the key definition for function key F2. This allows the user to press F2 to execute the previously saved PROC FSEDIT code. Thus, the user is not required to type the SAS programming statements needed to access the FSEDIT screen.

■ END the keys window when the user access key for the data entry screen has been defined. This function key is only available to the user for whom it was defined, thus, this is not a good solution for data entry systems that are used by multiple personnel.

```
KEYS <DMKEYS>
Command ---> END

Key       Definition

PF1       help
PF2       PGM ; INC 'A:\SUGIDEMO\FSEDIT.SAS' ; SUBMIT ;
PF3       zoom off; submit
PF4       pgm; recall
KP_0      help
KP_1      zoom off; submit
KP_2      pgm; recall
KP_3      forward
KP_4      left
KP_5      rfind
KP_6      right
KP_7      home
KP_8      rchange
KP_9      backward
KP_,      icon
KP_-      pmenu
KP_.      keys
Ctrl_A    next
                                                    ZOOM
```

9) Create User Access - Mutiple Users

■ Other options are available that would allow you to provide easy user access for a data entry system that is used by numerous people. This would include the option of setting up a Windows icon to place on the user's Windows desktop or creating a command file (CLIST, EXEC, SCRIPT, etc.) to invoke the system. In both cases, the method of invoking the data entry system would simply invoke SAS and specify an AUTOEXEC file to be executed at SAS initialization. The AUTOEXEC file would contain the code used to invoke the data entry application (Step 8), and then ENDSAS.

10) Use Your System

■ Press the function key or execute the command file associated with your application to start working !

■ Creating a 'tip sheet' for end users is recommended. This tip sheet would include instructions for logging into the data entry system, as well as helpful commands that are available for use while in PROC FSEDIT. The FSEDIT tips could be hard coded on the screen on a physical 'help screen' that would be part of the created screen.

■ Use the ADD and DUP commands to enter new observations into your data set.

■ Use FORWARD and BACKWARD to scroll through the observations in your data set. Use LEFT and RIGHT to scroll through the physical screens for an observation.

■ When attempting to enter data, but the terminal beeps at you and refuses your keystrokes, check to be sure that the terminal INSERT mode is off.

■ Use the FIND (F) command to find observations meeting specific criteria. Note that the searching moves forward from the current observation.
ex.
```
COMMAND ===> F DATE = .
```
to find observations with missing dates

■ Use the LOCATE (L) command to search forward for exact values of the variable specified as the Name Command Variable in the screen attributes.
ex.
```
COMMAND ===> L 12345678
```
to locate the next observation where ID = 12345678

■ Use the SEARCH (S) command to search forward for partial values of any of the variables specified as String Command Variables in the screen attributes.
ex.
```
COMMAND ===> S 12345678
```
to locate the next observation where any variable listed in the String Command Variables list contains the character string '12345678'.

■ Enter a number on the command line to go to the specified observation number. This is very useful to go to the first observation when you want to search forward using FIND, LOCATE, or SEARCH, and when going to the last observation
ex.
```
COMMAND ===>  1
```
to go to observation 1

```
COMMAND ===> 9999
```
to go to observation 9999 (or last observation if fewer than 9999 observations)

■ Use the WHERE command to subset the data set while in PROC FSEDIT.
ex.
```
COMMAND===> WHERE DATE = .
```
would subset the data set to allow editing of only those observations that have a missing DATE variable.

```
COMMAND ===>  WHERE
```
would reset any previously defined WHERE, allowing all observations to be edited.

Conclusion

As shown in this paper, PROC FSEDIT, combined with some minimal data management techniques, allows you to quickly and easily create custom data entry systems for use by non-SAS programmers.

References

Lajiness, Mic (1995), "An Introduction to the Power of PROC FSEDIT", *Proceedings of the Twentieth Annual SAS Users Group International Conference,* 385-390

SAS Institute, Inc. (1989), *SAS/FSP Software: Usage and Reference, Version 6, First Edition*, Cary NC: SAS Institute, Inc.

Acknowledgements

This paper is derived from a document produced for inclusion in the Data Resource and Utilization Manual, a programmer's reference guide, at the Kaiser-Permanente Division of Research, Oakland, CA.

Author's Address

Steven A. Wilson
Resource Biometrics, Inc.
5801 Christie Avenue Suite 355
Emeryville, CA 94608-1928
(510) 597-7217
e-mail: wilson@resbiom.com