

“FS-What?”

A New Use for an Old SAS Component

Kim Truett, KCT Data, Inc., Alpharetta, GA

ABSTRACT

Before AF/Frame and WebAF, before GUI meant pretty and colorful, and before Windows took over the world, SAS had FSEDIT®. This line-based “GUI” view of a SAS database has clever organization behind it and is quick to setup and efficient to use.

FSEDIT is much less used now, except in legacy systems, and this is unfortunate, because its simplicity and database driven design allows for very quick set-up and is very useful for some simple problem solving.

One use is for adding codes to a database. While this can be done 'blindly' using merges or SQL for exact matches, many terms need to be manually reviewed and the 'best code' selected. This paper utilizes a database of Adverse Events that needs codes. FSEDIT is used to display each event, then either look-up, or add in, coding information from a second database, just by typing in the first few letters of a proposed coding term or code and hitting return . . . But this isn't a talk about coding – it is about how to create a quick FSEDIT screen to simplify a database task.

Introduction

FSEDIT is a simple, line based editor for viewing and editing data. Now it is often overlooked by programmers drawn to the more powerful but more complex AF/Frame® and Web AF® application development. But sometimes just a screwdriver is needed, not a super-high-powered cordless drill with screwdriver attachment and 5 forward torques.

FSEDIT is the screwdriver. While it can be used for complex tasks, it is also simple, efficient, and doesn't require much novel programming.

The example presented here is coding clinical trials data, but FSEDIT can be used for many different tasks, so this paper focuses on how to build an FSEDIT tool, not on the tasks that this particular tool performs.

A. Invoking FSEDIT

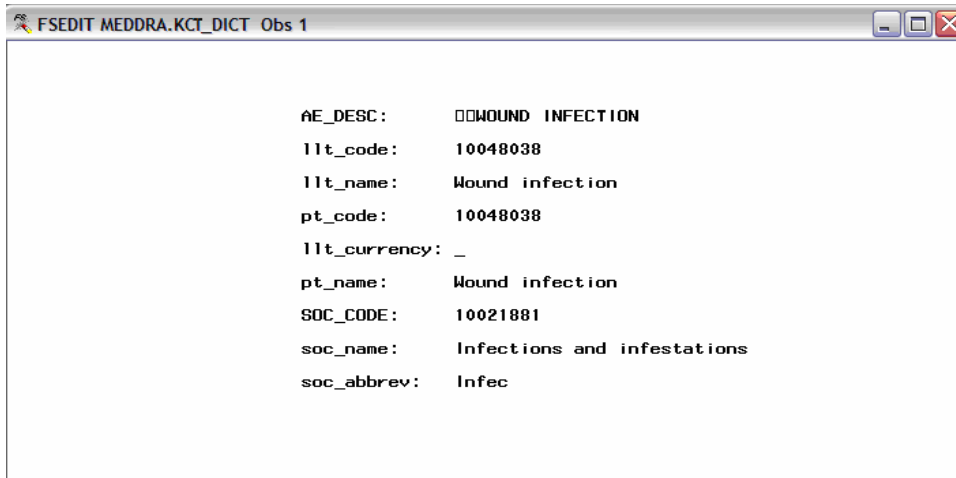
FSEDIT is a SAS procedure. So, like all SAS procedures it is invoked from a SAS editor.

```
proc fsedit
  data=libname.datasetname
  screen=libname.datasetname label;
```

The screen is not required to have the same name as the dataset, but keeping them the same helps with organization. Any other options used with data= in other procedures are also valid here. The LABEL option tells SAS to use that variable's label (instead of variable's name) in the FSEDIT window.

Sidebar: This can also be used to create a dataset. Replace data=libname.datasetname with new=libname.datasetname. If the new dataset is similar to an existing dataset, adding like=libname.otherdsname will copy the other dataset as the basis for the new dataset.

Submitting this line, brings up a window that looks like this:



AE_DESC: WOUND INFECTION
lit_code: 10048038
lit_name: Wound infection
pt_code: 10048038
lit_currency: _
pt_name: Wound infection
SOC_CODE: 10021881
soc_name: Infections and infestations
soc_abbrev: Infec

B. Customizing the FSEDIT screen

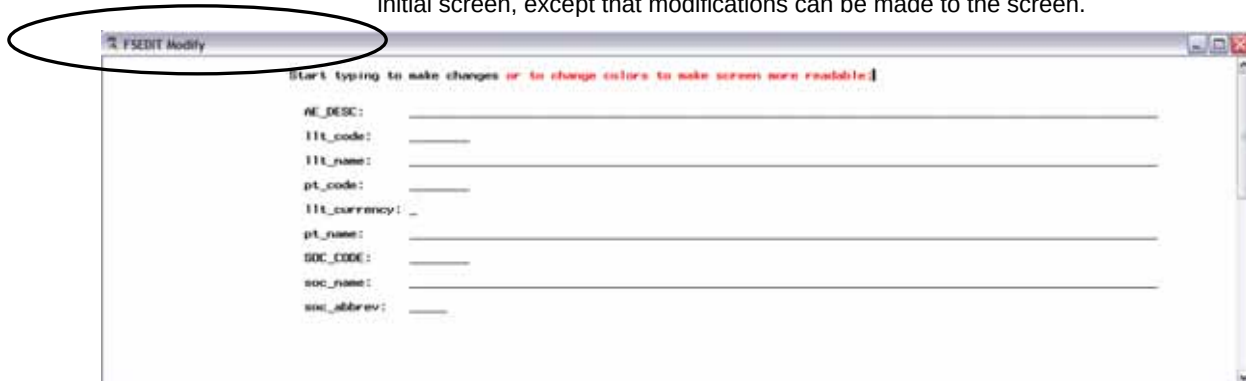
Generally, the first preference is to make this a bit more user friendly. All of the setup, design, coding and any other options are accessed by clicking the icon that looks like 'four squares' on the toolbar.



This brings up the following menu:

- 1 Information about screen modifications
- 2 Screen Modifications and Field Identification
- 3 Edit Program Statements and Compile
- 4 Assign Specific Attributes to Fields
- 5 Modification of General Parameters
- 6 Browse Program Statements

The first step involves choice 2 Screen Modifications and Field Identification. Clicking anywhere on this line (yes, on the line of text – this is pre-GUI) will bring up a screen that looks just like the initial screen, except that modifications can be made to the screen.



The image shows the FSEDIT Modify window. The title bar is circled. The window contains a list of variables and their values, similar to the initial screen, but with a red line indicating where to make changes. The variables are: AE_DESC, lit_code, lit_name, pt_code, lit_currency, pt_name, SOC_CODE, soc_name, and soc_abbrev.

This screen can now be changed into a more useful arrangement and titles and instructions can be added. The sidebar briefly describes the line editor commands. Lines represent places for data, and must be preceded by a blank (so ID__ won't work but ID ____ will). Lines are set by default to the variable length, but can be shortened or moved anywhere (copy and paste, or just retype). The color of the data displayed from the database will match the color of the lines so making the line blue makes the variable text blue.

Keeping the lines (where the dataset data will show up) a different color from the rest of the text on the screen improves the screen's clarity. The color of the text is changed by pressing [Esc] then typing a letter for each color (b=blue, r=red, o=orange, k=black) or by highlighting text and typing 'color mtext blue' (for example) on the command bar. Titles, instructions and any other text can be added to the screens here as well.

When the screen modification is complete, the variables from the database need to be 'relinked' to the lines on the screen. This is done by pressing F8 to exit the screen.

SAS asks if any calculated variables were created and then brings up the variable identification screen. The variable name appears in the bottom left corner. The variables are identified by placing the cursor on the appropriate line then pressing ENTER.

Sidebar: Line editor commands are useful to modify FSEDIT screens. First, type 'nums' in the command bar to bring up the line numbers. Then the following commands can be typed in those numbers to make changes:

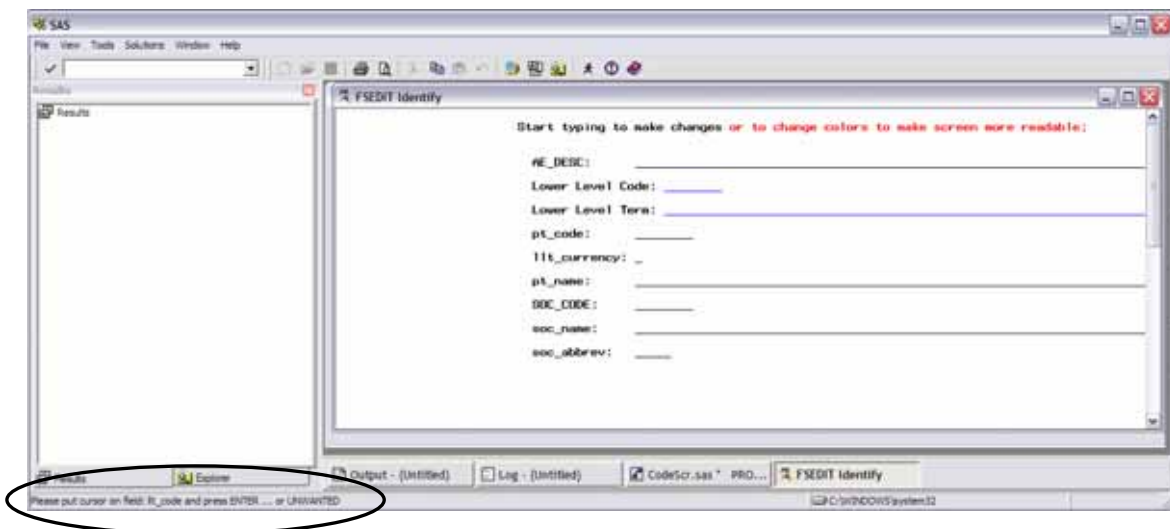
I(insert) – insert line (I3 to insert 3 lines) Also used in IA (insert after) IB (insert before).

M(move) to move a line, then B(before) or A(after) on the line to move it to its new location.

C(copy) to copy a line then B(before) or A(after) on the line to paste it to its new location.

D(delete) – delete the line

R(replicate) – replicate a line.



This is repeated for each line to associate it with the variable from the dataset. If the dataset has variables that are not needed on the screen, right click and select Tool-Unwanted. This does not delete the variable from the dataset, but simply removes it from the screen.

C. SCL basics for the FSEDIT screen

Now the screen can be used to review data within a database and to enter or modify data in a dataset. Nothing else is needed. But significant functionality can be added to the screen by adding a program behind it. To add a program behind the screen select 3 Edit Program Statements and Compile from the 'four squares menu'. This brings up a blank window ready for SCL code.

The code behind an FSEDIT screen is divided into sections that tell the screen when to execute that bit of code. Sections start with a keyword, followed by a colon and end with a RETURN;. The required sections are

```
FSEINIT:
RETURN;
```

```
INIT:
RETURN;
```

Sidebar: It is a good programming practice to set up all these sections in the FSEDIT SCL. Then placeholders are present for all sections, even if they are not used.

```
MAIN:
RETURN;
```

```
TERM:
RETURN;
```

```
FSETERM:
RETURN;
```

FSEINIT runs when the FSEDIT window is opened. FSETERM runs when the FSEDIT window is closed. These labeled sections do not rerun between records being displayed. Each time a new record is displayed, INIT runs. Within that record, every time ENTER is pressed, MAIN runs. When the user moves to a new record, TERM runs. Code that should run at the beginning of each record is placed in INIT, and code that should run as the record is exited is placed in TERM. Since the code in MAIN runs when ENTER is pressed, interactive code should be placed in MAIN.

After code is added, pressing F8 will compile the program and any error messages or a successful compilation message will appear in the log.

D. Example of SCL Code

This section will focus on three things that can be done with SCL code.

- A. Create a lookup function within it
- B. Link a secondary database
- C. Populate fields from a second database

```
IF LLT_NAME='?' THEN DO;                                /* If user types a '?' in LLT_NAME */
  SELDATA=OPEN('MEDDRA.LLT_DICT','I');                /* Open the dictionary database */
  LLT_NAME=DATALISTC(SELDATA,'LLT_NAME','PLEASE SELECT A TERM'); /* Create a datalist
                                                             Window */
  CALL SET(SELDATA);                                    /* When control is returned to the FSEDIT */
  RC=FETCH(SELDATA);                                   /* screen, fetch the record that the user */
  RC=CLOSE(SELDATA);                                   /* selected and close the database */
END;

ELSE IF LLT_NAME=' ' THEN DO;                            /* If user types a ' ' in LLT_NAME */
  LLT_CODE = .;                                         /* set all other coding variables to */
  PT_CODE = .;                                         /* missing - this deletes all coding data */
  PT_NAME = ' ';                                       /* from the record. */
  SOC_CODE = .;
  SOC_NAME = ' ';
  SOC_ABBREV = ' ';
END;

ELSE DO;                                                 /* Otherwise, */
  SELDATA=OPEN('MEDDRA.LLT_DICT','I');                /* Open the dictionary database */
  CALL SET(SELDATA);                                    /* Get the first record where the beginning */
  RC=WHERE(SELDATA,'UPCASE(LLT_NAME)=:'||QUOTE(LLT_NAME)); /* of the LLT_NAME matches */
  RC=FETCH(SELDATA);                                   /* the string just entered. */
  RC=CLOSE(SELDATA);                                   /* Fetch the record and close the database. */
END;
```

E. Changing the look and feel to make it 'designer'

Other options to modify the fields and screens and improve the user interface can be found under 4 Assign Specific Attributes to Fields and 5 Modification of General Parameters.

4 Assign Specific Attributes to Fields - Assigning attributes to fields allows control of items like initial value, maximum or minimum, whether a variable is required, and allows fields, such as ID field, to be protected so that they cannot be modified. The screen looks similar to the FSEDIT window, but at the top left of the screen, the attribute is listed, and on the bottom left are the choices for that attribute (if there are applicable choices).

FSEDIT Attribute: MAXIMUM

Coding Screen for Non-Match Terms

AE_DESC: _____

lit_name: _____

Code nothing below here - will autofill

pt_code: _____

lit_currency: _____

lit_code: _____

pt_name: _____

SOC_CODE: _____

soc_name: _____

soc_abbrev: _____

5 Modification of General Parameters - Modifying the general parameters allows modification of more general attributes such as the color of the screens. The most used parameters here are the modification of the background color, which is circled below, and the Window rows and Window cols, which set the size of the windows.

FSEDIT Params

Area	Color	Attribute
Background	GRAY	NONE
Border	SYSFORE	NONE
Header	SYSFORE	NONE
Command	SYSFORE	NONE
Message	RED	NONE
Protected	SYSFORE	NONE
Unprotected	SYSFORE	NONE

Keys name: FSEDIT

Override on errors: Y

Override on required: Y

Autosave value: 25

Modify password: _____

Name command variable: _____

String command variables: _____

Allow DELETE command: Y

Allow ADD/DUP command: Y

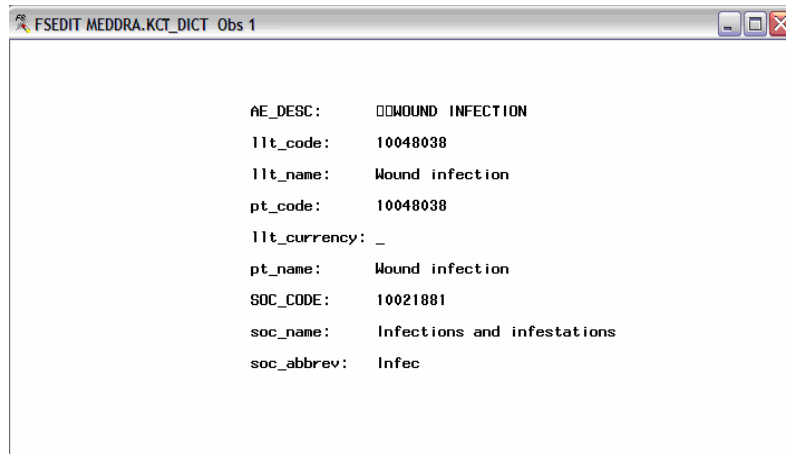
Window rows: 31

Window cols: 115

Start row: 1

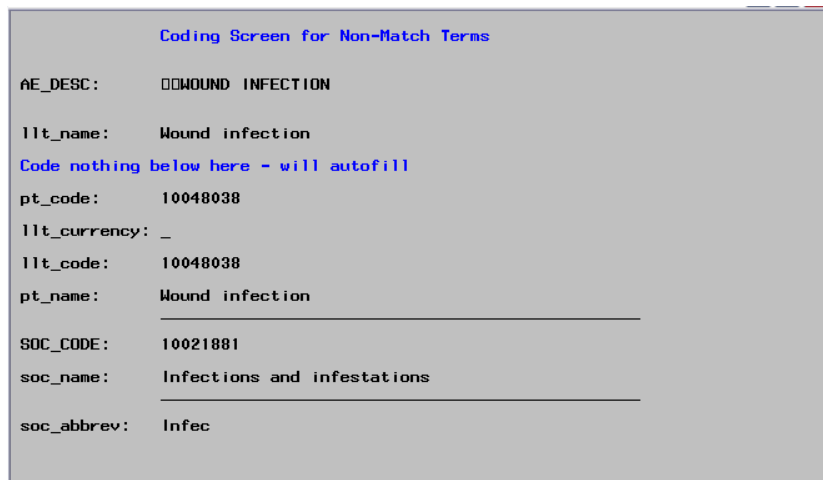
Start col: 1

The end result of clarifying the text, adding code, and changing the background color is shown below. Most of these changes (except for the SCL code) were cosmetic, but do add clarity to the final screen.



```
AE_DESC:      WOUND INFECTION
llt_code:     10048038
llt_name:     Wound infection
pt_code:      10048038
llt_currency: _
pt_name:      Wound infection
SOC_CODE:     10021881
soc_name:     Infections and infestations
soc_abbrev:   Infec
```

This was the initial screen:



```
Coding Screen for Non-Match Terms

AE_DESC:      WOUND INFECTION
llt_name:     Wound infection
Code nothing below here - will autofill
pt_code:      10048038
llt_currency: _
llt_code:     10048038
pt_name:      Wound infection
SOC_CODE:     10021881
soc_name:     Infections and infestations
soc_abbrev:   Infec
```

This is the final screen:

Conclusion

FSEDIT is a useful tool. This paper reviewed

- how to set it up
- how to create a lookup function within it
- how to link in a secondary database
- how to populate fields from a secondary database
- how to make it 'designer'

and should provide the basics for other simple data management tasks that require just a screwdriver.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Kim Truett
11585 Jones Bridge Road; Suite 420-115
Alpharetta, GA 30022
770.372.0989
Email: KCTData@comcast.net

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.