

Advanced Usage Techniques of PROC FSEDIT

Andrew P. Ford, Northern Indiana Public Service Company

ABSTRACT

This paper presents an advanced application of FSEDIT under version 6. This paper shows how to:

- * Link customized help screens, built with PROC BUILD, to an FSEDIT screen for both overall general help information as well as for individual entry field help.
- * Create customized commands.
- * Link a selection menu built with PROC PMENU with a FSEDIT screen.
- * Test for entry errors and display appropriate error messages.
- * Link together two FSEDIT screens to provide a tool for previewing output before printing a hard copy.
- * Link customized PF key settings for both FSEDIT and OUTPUT windows.

INTRODUCTION

A focus of the Electric Engineering Department is coordinating Capital Budget Projects. The tools needed to coordinate over 1100 projects include data entry, data tracking and gathering, and printing customized reports on an as-needed and monthly basis. SAS FSEDIT, version 6, was the tool chosen because it could provide all of the capabilities needed. SAS/FSEDIT was chosen for ease of maintenance, future needs, and compatibility issues in future releases of the SAS product. The new capability of having a compiled SCL program linked to the FSEDIT screen was the key to developing a robust FSEDIT application. A comprehensive system incorporating the FSEDIT product as the main driver was developed containing selection menus, customized commands, general and individual screen field help, and customized printing capabilities. Improvement in the FSEDIT product enables comprehensive verification of data entry fields with appropriate error messages. The selection menus were developed using PROC PMENU and are linked into the FSEDIT screen by the SCL associated with the screen. An undocumented "trick" that combines two SCL functions enables capturing the WHERE command then 1) storing the subset for the customized reports and 2) processing the WHERE command in the FSEDIT session. This application uses two FSEDIT screens linked together via MACRO code. This allows the user to specify a desired report in the first FSEDIT screen. In the second FSEDIT screen the user can view this report

before printing it out.

SCREEN CONTROL LANGUAGE(SCL)

A compiled program associated with the FSEDIT screen enables you to create a very robust editing environment. This was not possible in FSEDIT before version 6. SCL can have five main sections; 1) FSEINIT, 2) INIT 3) MAIN, 4) TERM, 5) FSETERM. The FSEINIT section executes only once, upon entering the FSEDIT application. The INIT section executes each time a new observation is displayed. With CONTROL ALWAYS on, the MAIN section is executed by the enter key or PF-key. The TERM section executes every time you leave an observation to view the next one. The FSETERM section executes only when you leave the FSEDIT application.

Warning: Do not use a password on the FSEDIT screen as a syntax error will lock you out from the SCL.

FSEINIT SECTION

The SCL excerpt below shows the contents of the FSEINIT section for this application.

```
000008 FSEINIT:
000010 CONTROL ALWAYS;
000012 CALL EXECCMDI('SETHelp
          SCREENS.MAIN.GENERAL.HELP');
000014 LENGTH WHERESTR $200 HEADING
          $200 HEADING2 $200 SORTBY $200
000015 FIELD $200 DEST $4;
000017 MSG_ = 'EDIT YELLOW FIELDS.';
000020 * Link customized selection
          menu to this FSEDIT screen;
000021 RC=PMENU('SCREENS.MAIN.options');
000023 * turn menus on for application;
000024 CALL EXECCMD('PMENU ON');
000026 FIELD='';
000027 SORTBY='';
000028 RETURN;
```

Each of the five sections starts with their appropriate name, semicolon, and end with RETURN;. Line 10 turns on CONTROL ALWAYS which enables the MAIN section to process all keyboard entries. Line 12 associates a customized help screen to this FSEDIT screen. Any time you press the HELP PF-key or type HELP on the command line will display the customized general help screen, SCREENS.MAIN.GENERAL.HELP. For line 12, SCREENS is the data set library reference name, MAIN is the catalog name, and GENERAL.HELP is the member name in the catalog. Line 14 defines some SCL variables which are utilized for the customized commands and reports. Line 21 links in the selection menu developed using PROC PMENU to the FSEDIT screen. Line 24

issues a display manager command that toggles on the selection menu from the command line.

INIT SECTION

```
000031 INIT:
000033   if loc_work='00' then
000034     do;
000035       alarm;
000036       cursor loc_work;
000037       erroron loc_work;
000038       MSG =
000039 'PLEASE ENTER VALID WORK
LOCATION DISTRICT NUMBER FOR
NEW PROJECT.';
```

```
000040   END;
```

```
000043 RETURN;
```

The INIT section checks a variable's value and if it is equal to 00, highlights the variable field in red, placing the cursor there, and writes a message below the command line or selection menu.

MAIN SECTION(This is where most of the processing logic resides)

CONTROL ALWAYS must be in effect to get the maximum processing of user keyboard inputs. The following are SOME excerpts from the MAIN section.

```
000049 MAIN:
000051 * SCL VARIABLE TO TEST FOR
CUSTOMIZED COMMAND;
000052 TESTCMD=WORD(1);
000054 * SCL VARIABLE TO TEST FOR
INDIVIDUAL HELP SCREENS;
000055 VARNAME=CURFLD();
000057 * SCL VARIABLE FOR WHERE STRING
PROCESSING;
000058 TYPE=WORDTYPE(1);
000060 * SCL VARIABLE FOR TESTING FOR
"ALSO" STRING IN WHERE COMMAND;
000061 CMD2=WORD(2);
000064 /** START OF TESTING FOR
INDIVIDUAL HELP SCREENS **/
000067 IF (TESTCMD='HELP' OR
TESTCMD='help') THEN
000068   DO;
000069     SELECT (VARNAME);
000070     WHEN ('CANCEL')
000071       DO;
000072       CALL EXECCMDI('SETHelp
SCREENS.MAIN.CANCEL.HELP');
000073       CALL EXECCMDI('HOME;
HELP');
000074       CALL EXECCMDI('SETHelp
SCREENS.MAIN.GENERAL.HELP');
000075       END;
000076       WHEN ('ITEM_NO')
000077         DO;
000078         CALL EXECCMDI('SETHelp
SCREENS.MAIN.ITEM_NO.HELP');
000079         CALL EXECCMDI('HOME;HELP');
000080         CALL EXECCMDI('SETHelp
SCREENS.MAIN.GENERAL.HELP');
000081         END;
000082         WHEN ('ACTSER')
```

```
000083       DO;
000084       CALL EXECCMDI('SETHelp
SCREENS.MAIN.ACTSER.HELP');
000085       CALL EXECCMDI('HOME;
HELP');
000086       CALL EXECCMDI('SETHelp
SCREENS.MAIN.GENERAL.HELP');
000087       END;
000190     OTHERWISE;
000191     END;
000192   END;
```

The above SCL code is the first part of the MAIN section. Lines 52, 55, 58, and 61 are instrumental in the processing done throughout this section. Line 52 assigns the SCL variable, TESTCMD, the first word typed on the command line. The delineator between words is a single blank. Line 55 assigns the SCL variable, VARNAME, the current location of the cursor. The SCL Variable TYPE on line 58 assigns the type of word that SCL variable TESTCMD represents. Line 61 assigns the SCL variable, CMD2, the second word typed on the command line. Line 67 starts the processing of the HELP command.

To receive individual help for an entry screen field move the cursor (Not the graphics cursor) to the entry field of interest and press the HELP PF-key. The select structure starting on line 69 determines which individual help screen to display. If no individual help screen is displayed then show the customized general help screen. After displaying the individual help screen the default help screen is reset to the customized general help.

```
000203 IF MODIFIED(LOC WORK) OR
MODIFIED(WORKSECT) THEN
000204   PROJ NO=WORKSECT || LOC WORK
|| SUBSTR(PROJ_NO,5,5) ;
000206 IF ( INDEX(WO_NO,'-') =0 ) THEN
000207   DO;
000208     ALARM;
000209     CURSOR WO_NO;
000210     ERRORON WO_NO;
000211     MSG =
000212 'DASH IS NOT ALLOWED TO BE
ENTERED FOR WORK ORDER NUMBER.';
000214   END;
```

The above section of SCL code shows the error checking abilities now available with FSEDIT. In line number 203, the MODIFIED function is an SCL function that indicates if a screen field has been changed or not. The SCL variables PROJ_NO, WORKSECT, AND LOC WORK correspond to the names of the screen field variables. The function INDEX in line 206 is one of the common functions available with base SAS. You can use any of these functions with the SCL. The ALARM statement sounds the audible

bell at the terminal. The CURSOR statement places the cursor at the screen field variable location WO NO. The ERRORON statement turns the screen field variable location to red.

```

000386 ***** HEADING COMMAND;
000387 IF (WORD(1)='HEADING' OR
      WORD(1)='heading') THEN
000388   DO;
000389   IF WORD(2)='SHOW' or
      word(2)='show' or
000390   WORD(2)='HELP' OR
      WORD(2)='help' THEN
000391     DO;
000392     IF WORD(2)='SHOW' OR
      WORD(2)='show' THEN MSG=HEADING;
00393     IF WORD(2)='HELP' OR
      WORD(2)='help' THEN
000394       DO;
000395       CALL EXECCMDI('SETHelp
      SCREENS.MAIN.HEADING.HELP');
000396       CALL EXECCMDI('HOME;
      HELP');
000397       CALL EXECCMDI('SETHelp
      SCREENS.MAIN.GENERAL.HELP');
000398       END;
000399       END;
000400     ELSE
000401       DO;
000402       IF WORD(2)='' THEN DO;
000403         HEADING='';
000404         MSG='TITLE HAS BEEN CLEARED';
000405       END;
000406       ELSE DO;
000407         IF HEADING='' THEN
000408           MSG='TITLE SPECIFIED';
000409         ELSE
000410           MSG='TITLE ADDED TO';
000411         HEADING=HEADING || ' ' ||
      WORD(2) || ' ' || WORD(3);
000412         CALL NEXTWORD();
000413         DO WHILE(WORD(3)!='')
      AND WORD(3)!=' ');
000414         HEADING=HEADING || ' ' ||
      WORD(3);
000415         CALL NEXTWORD();
000416       END;
000417     END;
000418   END;
000419 END;

```

The customized commands developed are for producing quick individual, customized reports. The customized commands enable the user to specify the first two title lines (HEADING and HEADING2), sort the listing (SORTBY), define which variables to print (FIELD), and subset the data(WHERE). The code for the HEADING command is shown above. The WHERE command is not an unique customized command. Lines 323 through 384 show an example of how to capture the WHERE command for customized reports and for processing the reserved WHERE command in the FSEDIT session.

Using the special keyword SHOW after the customized commands allows the current string values associated with those commands to be displayed on the message line. Using the keyword HELP after any of the special commands gives individual help concerning that command.

```

000323 ***** WHERE COMMAND ;
000324 IF (WORD(1)='WHERE' OR
      WORD(1)='where') THEN
000325   DO;
000326   IF WORD(2)='SHOW' or
      word(2)='show' or
000327   WORD(2)='HELP' OR
      WORD(2)='help' THEN
000328     DO;
000329     IF WORD(2)='SHOW' or
      word(2)='show' then do;
000330       MSG=WHERESTR;
000331       CALL NEXTCMD();
000332       CALL NEXTCMD(); CALL NEXTCMD();
000333     END;
000334     IF WORD(2)='HELP' OR
      WORD(2)='help' THEN
000335       DO;
000336       CALL EXECCMDI('SETHelp
      SCREENS.MAIN.WHERE.HELP');
000337       CALL EXECCMDI('HOME; HELP');
000338       CALL EXECCMDI('SETHelp
      SCREENS.MAIN.GENERAL.HELP');
000339     END;
000340   END;
000341   ELSE
000342     DO;
000343     if wherestr !='' and
      (cmd2='also' or cmd2='ALSO') then
000344       wherestr=wherestr || '&';
000345     IF WORD(2)='' THEN DO;
000346       wherestr='';
000347       MSG='WHERE SUBSETTING STRING
      HAS BEEN CLEARED';
000348     CALL EXECCMDI('WHERE','NOEXEC');
000349     END;
000350   ELSE DO;
000351     IF WHERESTR='' THEN
000352       MSG='WHERE
      COMMAND HAS BEEN SPECIFIED';
000353     ELSE
000354       MSG='WHERE
      COMMAND HAS BEEN ADDED TO';
000355     DO WHILE ((TYPE !=
      'EOD') AND (TYPE !='SEMI'));
000356     IF TYPE='STRING' THEN
000357       DO;
000358       IF INDEX(WORD(1),'') THEN
000359         WHERESTR=WHERESTR ||
      WORD(1) || ' ';
000360       ELSE
000361         WHERESTR=WHERESTR ||
      WORD(1) || ' ';
000362     END;
000363   ELSE DO;
000364     IF WHERESTR
      !='' AND (CMD2='ALSO' OR
      CMD2='also')

```

```

000366      AND ( WORD(1) ='WHERE' OR
              WORD(1)='where')
              THEN DO;
000368CALL NEXTWORD() CALL NEXTWOOD();
000370      END;
000371      if (word(1)
          ~= 'WHERE' and word(1) ~= 'where')
000372 or ((word(1)='WHERE' or
          word(1)='where') and
          wherestr='')
000374 then WHERESTR=WHERESTR || ' '
          || WORD(1);
000375      END;
000376      CALL NEXTWORD();
000377      TYPE=WORDTYPE(1);
000378      END;
000379 Wherestr=upcase(wherestr);
000380 DMS_CMD=TRIM(LEFT(WHERESTR));
000381 CALL EXECCMDI(DMS_CMD, 'NOEXEC');
000382      END;
000383      END;
000384      END;

```

The above SCL code from the MAIN section processes the WHERE command. This code is unique in that the reserved command WHERE uses both customized processing and normal FSEDIT processing. Lines 323 through 341 are used to process the HELP and SHOW special second commands as previously stated. Starting with line 343, the WHERE command is processed. The SCL variable WHERESTR is a string variable that will contain the correct WHERE statement syntax for use with "PROC PRINT" or "PROC FSEDIT PRINTALL". The SCL variable WHERESTR is continuously appended to when specifying each time the WHERE command. Both the FSEDIT WHERE processing and the SCL variable WHERESTR are reset when entering just the WHERE command on the command line. Line 377 and 381 demonstrates the two undocumented "tricks" to capture a reserved command and process it normally. The function WORDTYPE will return to you the type of any one of the first three tokens from the command line just like the WORD function returns to you the value of any one of the first three tokens. The syntax for the WORDTYPE function is like the WORD function and is shown on line 377. Values returned from the WORDTYPE function include the following:

SEMI - the semicolon. To distinguish between a ; ending a command and a ;' value being searched for.
NAME - a SAS name value (like the name of a variable or data set). Starts with a letter or underscore and contains only letters, numbers, or underscores.
STRING - anything the user places quotes around.
EOB - End of data. Tells us the

command didn't end with a semicolon but there are no other tokens on the command line.

NUMBER - a numeric value (possibly containing a '.', '-', or 'E').

INTEGER - a set of digits. Like a number with no special characters.

SPECIAL - operators used in expressions such as '=' or '>'.

There is a parameter you can place in the EXECMDI routine that executes the specified command without executing statements in the MAIN section of your SCL program. This is the NOEXEC parameter as illustrated on line 381. To capture the command from the command line and rebuild it in release 6, you can take advantage of the WORDTYPE SCL function and the NOEXEC option for the EXECMDI SCL routine as illustrated in the above SCL code, lines 352 through 384.

```

000508      ***** PRINTNOW COMMAND ;
000509 IF (WORD(1)='PRINTNOW' OR
      WORD(1)='printnow') THEN
000510 DO;
000512   If WORD(2) ='HELP' OR
      WORD(2)='help' THEN
000513   DO;
000514     CALL EXECCMDI('SETHelp
SCREENS.MAIN.printnow.HELP');
000515     CALL EXECCMDI('HOME; HELP');
000516     CALL EXECCMDI('SETHelp
SCREENS.MAIN.GENERAL.HELP');
000517     END;
000518   else do;
000521   if dest ='' then
000522     dest='SLC1';
000524     MSG='PRINT SUMMITTED'
000525 || '-LISTING PRINTED ' || DEST;
000527   IF FIELD='' THEN
000528   DO;
000529     SUBMIT CONTINUE;
000530     %LET TRYAGAIN='OK';
000531     %LET PRTDONE='';
000532     options ps=58 ls=132;
000546     PROC FSEDIT DATA=COPY
      PRINTALL NR=58
000547 screen=screens.main.print.screen;
000548     &wherestr;
000550     run;
000551     ENDSUBMIT;
000552     END;
000553   ELSE
000554   DO;
000555     SUBMIT CONTINUE;
000556     %LET TRYAGAIN='OK';
000557     %LET PRTDONE='';
000558     options ps=54 ls=132;
000570     PROC PRINT DATA=COPY ;
000571     &FIELD
000572     ;
000573     &wherestr
000574     ;
000575     run;
000576     ENDSUBMIT;

```

```

000577     END;
000579 IF HEADING2 ^='' THEN
000580 DO;
000581     SUBMIT CONTINUE;
000582     TITLE2 "&HEADING2 " ;
000584     RUN;
000585     ENDSUBMIT;
000586 END;
000588 DO;
000589     SUBMIT CONTINUE;
000590     TITLE2 "SAS LIBRARY:  ";
000592     RUN;
000593     ENDSUBMIT;
000594 END;
000596 IF HEADING ^='' THEN
000597 DO;
000598     SUBMIT CONTINUE;
000599     TITLE "&HEADING ";
000601     RUN;
000602     ENDSUBMIT;
000603 END;
000604 ELSE
000605 DO;
000606     SUBMIT CONTINUE;
000607     TITLE "ON-LINE PRINT ";
000609     RUN;
000610     ENDSUBMIT;
000611 END;
000613 IF SORTBY ^='' THEN
000614 DO;
000615     SUBMIT CONTINUE;
000616     PROC SORT DATA=COPY;
000617     &SORTBY
000618     ;
000619     RUN;
000620     ENDSUBMIT;
000624 end;

```

The customized command PRINTNOW, prints the data, outputting reports to the OUTPUT window. The block of SCL code processing the PRINTNOW command starts at line 508 and ends at line 624. Just like the other customized commands, you can get specific user help by typing the word HELP after the PRINTNOW command on the command line. This code to display the help screen is on lines 512 through 517. All other customized command specifications must have been done before issuing the PRINTNOW command. The values of the SCL variables associated with the customized commands (WHERESTR, SORTBY, HEADING, HEADING2, FIELD) are used within the processing of the PRINTNOW command, lines 508 to 624. The substitution of these values is done in this section of the code so that all of the other customized commands must have already been specified. Note the order the SCL processing is done for proper results of the report in the OUTPUT window. The SUBMIT CONTINUE-ENDSUBMIT blocks act as a FILO (First In Last Out) Que that release the code in these blocks for execution in the reverse order they are submitted.

SECOND FSEDIT SCREEN SCL CODE

```

000038 MAIN:
000060 ***** LANDSCAPE COMMAND
000061 IF (WORD(1)='land' OR
      WORD(1)='LAND') THEN
000062 DO;
000063     land='x';
000064     font = 'XCP14iso-L';
000065     marg = 'm510,40,0,50,660';
000066     lnsz = '132';
000067     pgsz = '54';
000068 end;
000071 ***** portrait COMMAND;
000072 IF (WORD(1)='PORT' OR
      WORD(1)='port') THEN
000073 DO;
000074     port='x';
000075     font = 'Titan10iso-P';
000076     marg = 'm660,30,0,45,510';
000077     lnsz = '75';
000078     pgsz = '54';
000079 end;
000081 ***** TRY AGAIN COMMAND;
000082 IF (WORD(1)='TRYAGAIN' OR
      WORD(1)='tryagain') THEN
000083 DO;
000084     SUBMIT CONTINUE;
000085     %LET TRYAGAIN='OK';
000086     %LET PRTDONE='';
000087     RUN;
000088     ENDSUBMIT;
000089 end;
000091 ***** PRTDONE COMMAND;
000092 IF (WORD(1)='PRTDONE' OR
      WORD(1)='prtdone') THEN
000093 DO;
000095     SUBMIT CONTINUE;
000096     dm output 'print file=out1
      append; clear';
000097     %LET TRYAGAIN='';
000098     %LET PRTDONE='X';
000099     RUN;
000106     data _null_;
000107         file _out1 print notitles;
000108         put '=UDK=';
000109         put '~+X';
000110         put 'Resetting this printer';
000112         proc printto; run;
000114     ENDSUBMIT;
000116     IF land ^='' or port ^=''
000117     THEN DO;
000118         SUBMIT CONTINUE;
000122         options nocaps ps=&pgsz
            ls=&lnsz;
000123         proc printto print=out1;
000124         data _null_;
000125             file _out1 print notitles;
000126             put '=UDK=';
000127             put "~+0&font";
000128             put "~0-zg-&marg";
000129         run;
000130         data _null_;
000131             file _out1 print;
000132             put "&font";
000133         run;
000134         ENDSUBMIT;
000135     end;

```

```

000137     END;
000140     RETURN;
The second FSEDIT screen uses the above
SCL MAIN section code, lines 38 through
140. The purpose of this FSEDIT screen
is to allow you a chance to preview the
output specified in the first FSEDIT
screen. This screen uses customized
PMENU selection menus. When specifying
a printer, the user can specify either
to set the laser printer to print in
landscape or portrait modes. Lines 60
through 68 set the necessary parameters
to have the printer set to landscape
mode. Lines 71 to 79 set the parameters
to have the printer set to portrait
mode. After you view the output and
indicate a printer location, the print
option can be selected. Lines 91
through 137 sends the contents of the
OUTPUT window to the desired printer in
the desired format.

```

MACRO CODE

```

000107 %MACRO CHOSESCR;
000108 %LET TRYAGAIN='';
000109 %LET PRTDONE='';
000110 DM 'CLEAR';
000111 PROC FSEDIT DATA=COPY
        SCREEN=SCREENS.MAIN.PROJ6.SCREEN;
000124 RUN;
000126 %MEND CHOSESCR;
000128 %CHOSESCR; RUN;
000132 %MACRO PRTSHOW;
000133     DATA TEMP2;
000134     LENGTH ASK $1.; RUN;
000136     DM 'CLEAR';
000137     PROC FSEDIT DATA=TEMP2
        SCREEN=SCREENS.MAIN.PRTSHOW.SCREEN;
000139     RUN;
000140 %MEND PRTSHOW;
000144 %MACRO SHOW;
000145     %DO %WHILE(&TRYAGAIN='OK');
000146         %PRTSHOW ;RUN;
000148         %IF &TRYAGAIN='OK' %THEN
000149             %DO;
000150                 TSO FREE F(OUT1);
000151                 %CHOSESCR; RUN;
000153             %END;
000154     %END;
00155 %MEND SHOW;
000157 %SHOW

```

The two separate FSEDIT screens described when linked through the above macro code provides a method for the user to specify his customized report(s). The second FSEDIT screen, *prtshow.screen*, views the output through the OUTPUT window, and you decide if/where the hard copy report should be routed. The destination of the printed report can be any of the hundreds of local printers as well as the computer. center laser printer depending on size and desires of the user. The user can also, with macro variable TRYAGAIN, loop back to the

report specification FSEDIT screen, *proj6.screen*, to re-specify the report if the first report did not look as it should. Note that the macro variable TRYAGAIN is set to the appropriate value depending on what the user chooses to do in the SCL programs in both of the FSEDIT screens.

CHANGING OUTPUT WINDOW PF-KEYS

```

000019 LIBNAME SCREENS
        'GE.SAS.CATALOG2.SCREENS'
        DISP=SHR; RUN;
000021 PROC CATALOG C=SCREENS.MAIN
        ENTRYTYPE=KEYS;
000022 COPY OUT=SASUSER.PROFILE;
000023 SELECT OUTPUT; RUN;
000026 PROC CATALOG C=SASUSER.PROFILE
        ENTRYTYPE=KEYS;
000027 DELETE DMKEYS;
000028 CHANGE OUTPUT = DMKEYS ;RUN;
000031 ENDSAS;

```

The above code runs before entering the first FSEDIT screen. The purpose is to copy the customized PF-key definitions into the users sasuser.profile catalog so that the OUTPUT window PF-keys are the same as in the FSEDIT screen. This code must be run in a manner so that it is run BEFORE and SEPARATELY from the FSEDIT SAS session.

LINKING PF-KEYS WITH FSEDIT SCREEN

To link customized PF-keys to the FSEDIT screen type MOD on the command line while viewing the FSEDIT screen. Choose option 5. Type the name of the PF-keys catalog member next to KEYS NAME: and issue the end command. The .KEYS catalog member can be changed by using PROC BUILD. The statement, "Proc build c= screens.main ", will allow you to change, delete, add any member of the MAIN catalog. The customized keys member, .KEYS, should be in the same catalog as the FSEDIT screen.

REFERENCES

SAS Institute Inc., *SAS Procedures Guide, Version 6, Third Edition*, Cary, NC:SAS Institute., 1990, pp 433-461.

SAS Institute Inc., *SAS Screen Control Language: Reference, Version 6, First Edition*, Cary, NC:SAS Institute Inc., 1990.

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

SAS, SAS/FSP, and SAS/FSEDIT are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.