SAS/FSP® 9.1 Procedures Guide

The Power to Know®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2004. *SAS/FSP® 9.1 Procedures Guide*. Cary, NC: SAS Institute Inc.

SAS/FSP® 9.1 Procedures Guide

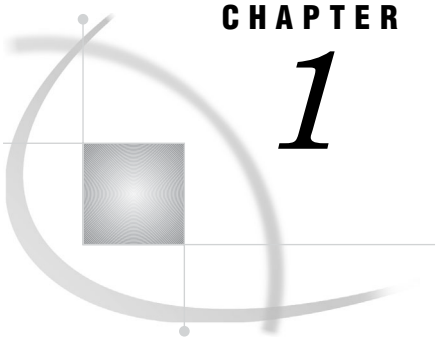
ISBN 1-59047-226-8

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, January 2004

SAS Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/pubs** or call 1-800-727-3228.

Contents



CHAPTER

1

Introduction to SAS/FSP Procedures

Overview

The procedures in SAS/FSP software provide convenient interactive facilities for data entry, editing, and retrieval. Using SAS/FSP software you can

- browse and edit the contents of SAS data sets
- enter data into existing SAS data sets
- create new SAS data sets
- browse and edit SAS data views created with SAS/ACCESS software
- browse SAS data views created with the SQL procedure in Base SAS software
- create, edit, and print form letters and reports
- build and customize end-user applications.

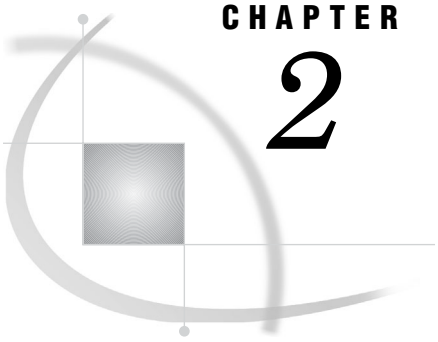
The following chapters provide descriptions and the full syntax of the procedures in SAS/FSP software and of the commands that are specific to each procedure.

Uses and Features of SAS/FSP Procedures

The following table shows which tasks each SAS/FSP procedure performs and what special features it provides.

Procedure	*Uses and Features*
FSBROWSE	*Uses* □ Browsing existing SAS data sets. □ Editing and updating existing SAS data sets. □ Creating new SAS data sets. □ Creating data entry and editing applications. *Features* □ Displays one observation at a time. □ Window characteristics can be customized. □ Application-specific key definitions can be assigned. □ Associated SAS Component Language programs can display computed variables, validate field values, and manipulate values from other SAS data sets. □ The FSLETTER procedure can be called to create letters or reports containing information from the displayed data set.
FSEDIT	*Uses* □ Browsing existing SAS data sets. □ Creating data presentation application. *Features* □ Displays one observation at a time. □ Window characteristics can be customized. □ Application-specific key definitions can be assigned. □ Associated SAS Component Language programs can display computed variables and values from other SAS data sets. □ The FSLETTER procedure can be called to create letters or reports containing information from the displayed data set.

Procedure	***Uses and Features***
FSLETTER	*Uses* □ Creating individual letters and reports. □ Creating personalized form letters and reports that incorporate values from a SAS data set. *Features* □ Customized forms can be used to define printer characteristics. □ Text-field highlighting can be used to take advantage of all available printer capabilities. □ The text editor's spelling checker can be used to verify words in entered text.
FSVIEW	*Uses* □ Browsing existing SAS data sets. □ Editing and updating existing SAS data sets. □ Creating new SAS data sets. *Features* □ Displays observations in tabular format. □ Window characteristics can be customized. □ Associated formulas can display computed variables and can validate or otherwise manipulate column values.



CHAPTER

2

The FSBROWSE Procedure

Overview

The FSBROWSE procedure opens the FSBROWSE window, from which you can browse the contents of a SAS data set one observation at a time. The FSBROWSE procedure is identical to the FSEDIT procedure, except that the FSBROWSE window does not allow you to make changes to the displayed data set.

The procedure provides the tools for building applications that display data. An FSBROWSE application provides a custom display in which you can specify how values are presented and also add descriptive text.

The FSBROWSE procedure also allows you to call the FSLETTER procedure from within an FSBROWSE session. This enables you to create form letters or reports that are personalized with information from the observations that are displayed by the FSBROWSE procedure.

Note: You can also open the FSBROWSE window by issuing an FSBROWSE command from any SAS System command line. △

FSBROWSE Procedure Syntax

Requirement: The FSBROWSE procedure must have an input data set. By default, the procedure uses the most recently created data set as its input data set. You can use the DATA= option in the PROC FSBROWSE statement to select a particular data set. If you do not specify a data set and none has previously been created in the current SAS session, the procedure terminates with an error message.

PROC FSBROWSE <DATA=*data-set*>
<KEYS=*keys-entry*>
<SCREEN=*SAS-catalog* <.*screen-entry*>> | <*display-options*>
<*procedure-options*>
<*letter-options*>;

where

- *display-options* can be one or more of the following:

  LABEL

  NC=*n*

  NOBORDER

  NR=*n*

  STCOL=*n*

  STROW=*n*

  TAB=*n*

- *procedure-options* can be one or more of the following:

  DEBUG

  MODIFY | MOD

  OBS=*n*

  PRINTALL

- *letter-options* can be one or more of the following:

  LETTER=*SAS-catalog*<.*letter-entry*> <SEND=*letter-entry*>

  PRINTFILE=*fileref*

FORMAT *variable-list format* <*... variable-list-n format-n*>;

LABEL *variable='label'* <*... variable-n='label-n'*>;

VAR *variable* <*... variable-n*>;

WHERE *expression*;

See "FSEDIT Procedure Syntax" on page 10 for detailed descriptions of the statements and options that are available to both the FSBROWSE and FSEDIT procedures. Exceptions for the FSBROWSE procedure are noted in the FSEDIT statement and option descriptions.

FSBROWSE Command Syntax

Tip: The FSBROWSE command does not allow you to specify procedure options such as KEYS=. You must use a PROC FSBROWSE statement rather than the FSBROWSE command to initiate the procedure with these options. You must also use the PROC FSBROWSE statement if you want to modify the procedure's behavior using the FORMAT, LABEL, VAR, or WHERE statements.

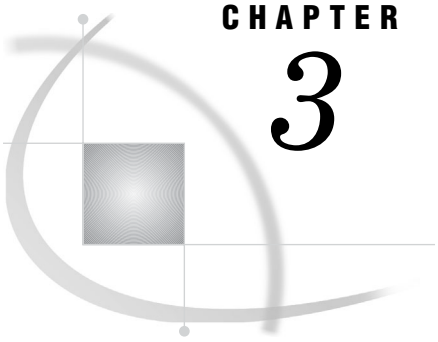
Tip: You must use the PROC FSBROWSE statement to produce letters or other documents from the FSBROWSE session because the FSBROWSE command does not provide a substitute for the statement's LETTER= option.

FSBROWSE <? | *data-set* <*screen-name*>>

See "FSEDIT Command Syntax" on page 21 for detailed descriptions of the arguments you can use with both the FSBROWSE and FSEDIT commands.

Using the FSBROWSE Procedure

The FSBROWSE procedure provides FSBROWSE versions of the same windows as the FSEDIT procedure (except for the FSEDIT NEW window). The same commands can be used in the FSBROWSE windows, except for commands that relate to editing the values of variables. Refer to "FSEDIT Window Commands" on page 26 for descriptions of the commands that are available in both FSBROWSE and FSEDIT windows. Exceptions for FSBROWSE windows are noted in the FSEDIT command descriptions.



CHAPTER

3

The FSEDIT Procedure

Overview

The FSEDIT procedure enables you to edit a SAS data set one observation at a time. You can also use it to create a new SAS data set.

The procedure provides the tools for building applications for entering and editing data. An FSEDIT application provides a custom display in which each data entry field has a set of attributes that can

- assign an initial value to the field
- restrict the range of values that can be entered in the field
- protect the field from editing
- require that a value be entered in the field.

The applications can also include a SAS Component Language (SCL) program that

- performs sophisticated error-checking and validation of values that are entered in the variable fields
- displays computed values in special fields
- manipulates values in other SAS data sets.

The FSEDIT procedure also enables you to call the FSLETTER procedure from within an FSEDIT session. This enables you to create form letters or reports that are personalized with information from the observations that are displayed by the FSEDIT procedure.

Note: You can also open the FSEDIT window by issuing an FSEDIT command from any SAS System command line. △

CAUTION:

The FSEDIT procedure edits a data set in place. The FSEDIT procedure does not leave an unedited copy of the original. If you need to preserve a copy of the original data, be sure to make a copy of the data set before you begin editing. △

FSEDIT Procedure Syntax

Restriction: Do not use any of the other statements when you use the NEW= option in the PROC FSVIEW statement. (The VAR statement causes an error; the FORMAT, INFORMAT, LABEL, and WHERE statements are ignored.)

PROC FSEDIT <DATA=*data-set* | NEW=*data-set* <LIKE=*data-set*>>
<KEYS=*keys-entry*>
<SCREEN=*SAS-catalog*<.*screen-entry*>> | <*display-options*>
<*procedure-options*>
<*letter-options*>;

where

- *display-options* can be one or more of the following:

  LABEL

  NC=*n*

  NOBORDER

  NR=*n*

  STCOL=*n*

  STROW=*n*

  TAB=*n*

- *procedure-options* can be one or more of the following:

  ADD | NOADD

  DEBUG

  MODIFY | MOD

  NODEL

  OBS=*n*

  PRINTALL

- *letter-options* can be one or more of the following:

  LETTER=*SAS-catalog* <.*letter-entry* > <SEND=*letter-entry*>

  PRINTFILE=*fileref*

FORMAT *variable-list format* <*... variable-list-n format-n*>;

INFORMAT *variable-list informat* <*... variable-list-n informat-n*>;

LABEL *variable='label'* <*... variable-n='label-n'*>;

VAR *variable* <*... variable-n*>;

WHERE *expression*;

The PROC FSEDIT statement is required. The other statements are optional and are used as follows:

To do this	Use this statement
Associate formats with variables in the input data set	FORMAT
Associate informats with variables in the input data set	INFORMAT
Assign labels that can be used in place of variable names to identify fields in the display	LABEL
Select which variables are available to the procedure	VAR
Specify a condition or set of conditions that observations in the input data set must meet in order to be processed	WHERE

PROC FSEDIT Statement

Initiates the FSEDIT procedure.

Requirement: The FSEDIT procedure must have an input data set. By default, the procedure uses the most recently created data set as its input data set. You can use the DATA= option in the PROC FSEDIT statement to select a particular data set. If you do not specify a data set and none has previously been created in the current SAS session, the procedure terminates with an error message.

Tip: Use the SCREEN= option to identify a SCREEN entry in which to store custom features of the FSEDIT session.

Note: The LETTER= option is required if you want to generate letters and other documents using the FSLETTER procedure within the FSEDIT session.

PROC FSEDIT <DATA=*data-set* | NEW=*data-set* <LIKE=*data-set*>>
<KEYS=*keys-entry*>
<SCREEN=*SAS-catalog*<.*screen-entry*>> | <*display-options*>
<*procedure-options*>
<*letter-options*>;

where

display-options provide control over the appearance of the FSEDIT window. All of the following options except NOBORDER and NR= are ignored if an existing SCREEN entry is specified with the SCREEN= option.

LABEL

NC=*n*

NOBORDER

NR=*n*

STCOL=*n*

STROW=*n*

TAB=*n*

procedure-options can be one or more of the following:

ADD | NOADD

DEBUG

MODIFY | MOD

NODEL

OBS=*n*

PRINTALL

letter-options enable you to generate letters, reports, and other documents during an FSEDIT session—things that you can also do independently with the FSLETTER procedure.

LETTER=*SAS-catalog* <.*letter-entry*>

PRINTFILE=*fileref*

SEND=*letter-entry*

Display and letter options are marked as such in the option descriptions that follow.

Options

Note: Most of the options that are listed here for the FSEDIT procedure are also available for the FSBROWSE procedure. However, the FSEDIT options that relate to creating and editing data sets are not applicable to the FSBROWSE procedure. △

ADD
creates a new blank observation when the procedure is initiated. The new observation is displayed for editing when the FSEDIT window is opened.

DATA=*data-set* <(*data-set-options*)>
names an existing SAS data set to be edited. By default, the FSEDIT procedure uses the most recently created data set.

If you specify both the DATA= option and the NEW= option in the same PROC FSEDIT statement, the DATA= option is ignored.

You can add a list of data set options following the data set name. The list must be enclosed in parentheses. Refer to *SAS Language Reference: Dictionary* for a listing and descriptions of data set options.

Note: The FSEDIT procedure ignores the data set options FIRSTOBS= and OBS=. All other data set options are valid. △

DEBUG
turns on the SAS Component Language (SCL) source-level debugger, which provides step-by-step assistance in resolving errors in SCL programs. This option is useful when you are creating or modifying an application that includes an SCL program.

See *SAS Component Language: Reference* for information about the SCL debugger.

KEYS=*keys-entry*
names the KEYS entry to be associated with the FSEDIT session. The KEYS entry contains function key assignments for the FSEDIT window.

Note: The KEYS= option is ignored when the SCREEN= option is also used in the PROC FSEDIT statement, unless a new SCREEN entry is being created. △

If you specify an existing SCREEN entry with the SCREEN= option, the KEYS entry name that is recorded in the SCREEN entry takes precedence over the entry that is specified in the KEYS= option.

The *keys-entry* value must be a one-level name. The search sequence for the specified entry is as follows:

1. If you also supply the SCREEN= option with the PROC FSEDIT statement, the procedure looks in the catalog named in that option for an entry that has the specified name and the type KEYS.
2. If the KEYS entry is not found in the catalog that contains the SCREEN entry, or if the SCREEN= option is not supplied, the procedure looks for an entry that has the specified name and the type KEYS in the SASUSER.PROFILE catalog (or in WORK.PROFILE if the SASUSER library is not allocated).
3. If the KEYS entry is not found in your personal PROFILE catalog, the procedure looks for an entry that has the specified name and the type KEYS in the SASHELP.FSP catalog.
4. If the KEYS entry is not found in the SASHELP.FSP catalog, the procedure searches the same sequence of catalogs for the default entry, FSEDIT.KEYS.

LABEL

(display option; ignored when an existing SCREEN entry is specified with the SCREEN= option)

specifies that variable labels, rather than variable names, are used to identify variable fields in the FSEDIT window. If a variable has no associated label, the variable name is used to identify that variable's field.

LETTER=*SAS-catalog*<*.letter-entry*>

(letter option)

names a SAS catalog that contains LETTER entries, or produces copies of a specified document for all observations in the data set. If the specified catalog does not exist, it is created.

The behavior of the FSEDIT procedure depends on which form of the option you use:

- If you specify only the catalog, the procedure does not automatically produce copies of a document (unless the SEND= option is also used). This form of the option makes the EDIT, LETTER, and SEND commands available during the FSEDIT session; these commands enable you to create and print copies of a document for individual observations.

  The general form of the *SAS-catalog* value is

  <*libref.*>*catalog-name*

  If you specify only a one-level name, it is treated as a catalog name in the default library, WORK. You must specify a two-level catalog name if you want to specify a letter name.

- If you specify a letter name in addition to the catalog name, a copy of the specified document is produced for each observation in the input data set. (If a WHERE statement is used in conjunction with the PROC FSEDIT statement, then copies are produced only for observations that satisfy the specified criteria.) In this case, the procedure does not open an FSEDIT window. A pause occurs while the copies are produced; then the procedure terminates.

  The general form of the *letter-entry* value is

  entry-name<.LETTER>

  If you specify a two-level letter name with anything other than LETTER as the second level (entry type), the specified entry type is ignored; LETTER is used instead.

You must specify an existing LETTER entry. The procedure terminates with an error message if the specified LETTER entry does not exist.

LIKE=*data-set* <(*data-set-options*)>
names an existing SAS data set whose structure is copied when a new SAS data set is created. (This option must be used in conjunction with the NEW= option.) When the FSEDIT NEW window is opened, the variable names and attributes of the data set that is specified in this option are displayed.

You can add a list of data set options following the data set name. The list must be enclosed in parentheses. Refer to *SAS Language Reference: Dictionary* for a list and descriptions of data set options.

MODIFY
MOD
opens the FSEDIT Menu window before opening the FSEDIT window. From the FSEDIT Menu window, you can perform tasks that modify the appearance and behavior of the FSEDIT window.

The MODIFY option is ignored (and a warning message is generated) if the PROC FSEDIT statement also includes the SCREEN= option specifying an existing SCREEN entry that is protected by a password.

NC=*n*
(display option; ignored when an existing SCREEN entry is specified with the SCREEN= option)

specifies the width in columns of the FSEDIT window. By default, the FSEDIT window occupies the maximum number of columns that is supported by the output device. The value of *n* must be at least 35. If you specify a value that exceeds the maximum number of columns available on your device, the procedure sets the window width to the maximum available width; no warning message is generated.

NEW=*data-set* <(*data-set-options*)>
creates a new SAS data set. The procedure terminates with an error message if a data set that has the specified name already exists.

When this option is used, the FSEDIT procedure begins by opening the FSEDIT NEW window, in which the names and attributes of the variables in the new data set are defined. Use the LIKE= option in conjunction with the NEW= option to initialize the FSEDIT NEW window that has the variable names and attributes of an existing data set. After the structure of the new data set is defined, the FSEDIT window is opened so that observations can be added. For details, see “Creating a New Data Set” on page 37.

You can add a list of data set options following the data set name. The list must be enclosed in parentheses. Refer to *SAS Language Reference: Dictionary* for a list and descriptions of data set options.

NOADD
prevents users from adding new observations to the data set. When you use the NOADD option, the ADD and DUP commands are disabled in the FSEDIT window, but users can still edit existing observations. When you use the NOADD option in conjunction with the SCREEN= option, the NOADD option takes precedence over the parameter setting that is specified in the SCREEN entry.

The FSEDIT procedure also provides a parameter that you can use to disable the ADD and DUP commands. Refer to “Modifying General Parameters” on page 57 for details.

NOBORDER
suppresses the sides and bottom of the FSEDIT window’s border in a character-based display environment.

Note: This option is ignored in graphical windowing environments. △

When this option is used in a supported display environment, text and fields can appear in the columns and row that the border normally occupies.

When the NOBORDER option is used in conjunction with the SCREEN= option, the window size that is specified in the SCREEN entry is ignored. The FSEDIT window always occupies the maximum possible number of rows and columns when the NOBORDER option is specified.

NODEL
prevents users from deleting observations from the data set. When you use the NODEL option, the DELETE command is disabled in the FSEDIT window, but users can still edit existing observations. When you use the NODEL option in conjunction with the SCREEN= option, the NODEL option takes precedence over the parameter setting that is specified in the SCREEN entry.

The FSEDIT procedure also provides a parameter that you can use to disable the DELETE command. Refer to "Modifying General Parameters" on page 57 for details.

NR=*n*
(display option)

specifies the height in rows of the FSEDIT window. By default, the FSEDIT window occupies the maximum number of rows that is supported by the output device. The value of *n* must be at least 10. If you specify a value that exceeds the maximum number of rows available on your device, the procedure sets the window height to the maximum available height; no warning message is generated.

Note: When the NR= option is used in conjunction with the SCREEN= option, the window size that is specified in the NR= option overrides the number of rows that is specified in the SCREEN entry. △

OBS=*n*
specifies the number of the observation that is displayed when the FSEDIT window is opened. By default, the first observation in the data set (observation 1) is initially displayed. If the *n* value is greater than the number of observations in the input data set, then the last observation in the data set is displayed when the FSEDIT window is opened.

This option is not valid if a WHERE statement is used with the PROC FSEDIT statement.

PRINTALL
prints a copy of the FSEDIT display for each observation in the data set. (If a WHERE statement is used in conjunction with the PROC FSEDIT statement, then only observations that meet the specified criteria are printed.) If the SCREEN= option is also specified, the custom display format is used.

The procedure output is written to the location that has been designated for SAS System output. If you are using the SAS windowing environment, the output is written to the OUTPUT window.

When you use the PRINTALL option, the procedure does not open an FSEDIT window. A pause occurs while the output is created; then the procedure terminates.

PRINTFILE=*fileref*
PRTFILE=*fileref*
PRINT=*fileref*
DDNAME=*fileref*
(letter option)

names an external file to which documents that are produced during the FSEDIT session are written. By default, output is sent to the output destination that is specified in the FORM entry that is associated with the LETTER entry. When this option is used, output is written to the specified file instead.

You must use a FILENAME statement to assign the fileref to an external file before submitting a PROC FSEDIT statement that contains this option.

SCREEN=*SAS-catalog*<*.screen-entry*>
names a catalog or a specific SCREEN entry that contains information for a custom FSEDIT application, or in which the procedure can store custom features that are defined during the current session. If the specified catalog does not already exist, it is created.

The general form of the *SAS-catalog* value is

<*libref*.>*catalog-name*

If you specify only a one-level name, it is treated as a catalog name in the default library, WORK. You must use a two-level catalog name if you want to specify a SCREEN entry name.

The general form of the *screen-entry* value is

entry-name<.SCREEN>

If you specify a two-level screen name that has anything other than SCREEN as the second level (entry type), the specified entry type is ignored; SCREEN is used instead.

When the SCREEN= option is used, the procedure attempts to load a SCREEN entry when the FSEDIT session is initiated:

- If only the catalog name is provided, the procedure attempts to load an entry that is named FSEDIT.SCREEN.
- If both the catalog name and the entry name are provided, the procedure attempts to load the specified entry.

If the entry is not found, the FSEDIT session is initiated that has default FSEDIT window characteristics.

If you do not supply the SCREEN= option, any changes that you make to the display format are available only during the current FSEDIT session.

SEND=*letter-entry*
(letter option)

generates a copy of the specified document for each observation in the input data set. (If a WHERE statement is used in conjunction with the PROC FSEDIT statement, then letters are generated only for observations that meet the specified criteria.) This option is valid only when the LETTER= option is also used.

The *letter-entry* value must be the name of an existing LETTER entry in the catalog that is identified in the LETTER= option. The value should be a one-level name; the entry type LETTER is assumed. The procedure terminates with an error message if the specified entry does not exist.

When you specify the SEND= option, the procedure does not open an FSEDIT window. A pause occurs while the copies of the document are produced; then the procedure terminates.

STCOL=*n*
(display option; ignored when an existing SCREEN entry is specified with the SCREEN= option)

specifies the display column in which the leftmost column of the FSEDIT window is positioned. By default, the FSEDIT window begins at the leftmost column of the display (STCOL=1).

STROW=*n*
(display option; ignored when an existing SCREEN entry is specified with the SCREEN= option)

specifies the display row in which the top row of the FSEDIT window is positioned. By default, the window begins at the top row of the display (STROW=1).

TAB=*n*
(display option; ignored when an existing SCREEN entry is specified with the SCREEN= option)

specifies the interval that is used for column spacing when more than one column is necessary to display variables in the default screen format.

FORMAT Statement

Associates formats with variables in the input data set. *Formats* are patterns that the SAS System uses to determine how variable values are displayed. The formats can be either SAS formats or custom formats that you have defined with the FORMAT procedure.

Reminder: FORMAT statements are ignored if you use them in conjunction with a PROC FSEDIT statement that includes the NEW= option.

FORMAT *variable-list format* <*... variable-list-n format-n*>;

Arguments

At least one pair of the following arguments is required:

variable-list
consists of one or more variable names from the input data set.

format
is the SAS format or user-defined format to associate with the specified variable or variables.

Using the FORMAT Statement

You can use a single FORMAT statement to assign the same format to several variables or to assign different formats to different variables. You can use any number of FORMAT statements with each PROC FSEDIT statement.

Formats that are specified in a FORMAT statement take precedence over formats that are defined in the data set itself. Formats that are assigned in a FORMAT statement remain in effect only for the duration of the procedure. The FORMAT statement does not affect any format assignments that are stored in the data set.

If you are creating a new application, or if you do not use a custom FSEDIT display, then the format widths that you specify may affect the widths of the fields for the associated variables in the FSEDIT window. If you are using an existing application, the assigned formats determine how variable values are displayed, but they do not affect field widths in the FSVIEW window.

Be aware that the format you assign to a variable affects the informats you can assign with the INFORMAT statement. For example, suppose the data set that is displayed by the FSVIEW procedure contains a variable AMOUNT that is assigned the format DOLLAR10.2 but an informat of 10.2. Because of the format, values in the column for the variable AMOUNT are displayed with commas and a leading dollar sign, so the value 1250 would be displayed as $1,250.00. However, if you edit this value (for example, changing it to $1,150.00) and press ENTER, an error condition occurs. The

10.2 informat does not allow the dollar sign ($) or comma characters in entered values. An appropriate informat for this variable is COMMA., which does allow these characters.

Refer to *SAS Language Reference: Dictionary* for more detailed information about formats and on the FORMAT statement. See the description of the FORMAT procedure in *Base SAS Procedures Guide* for information about defining your own formats.

INFORMAT Statement

Associates informats with variables in the input data set. *Informats* are patterns that the SAS System uses to determine how values that are entered in variable fields should be interpreted. The informats can be either SAS informats or custom informats that you have defined with the FORMAT procedure.

Reminder: INFORMAT statements are ignored if you use them in conjunction with a PROC FSEDIT statement that includes the NEW= option.

INFORMAT *variable-list informat* <*... variable-list-n informat-n*>;

Arguments

At least one pair of the following arguments is required:

variable-list
consists of one or more variable names from the input data set.

informat
is the SAS informat or user-defined informat to associate with the specified variable or variables.

Using the INFORMAT Statement

You can use a single INFORMAT statement to assign the same informat to several variables or to assign different informats to different variables. You can use any number of INFORMAT statements with each PROC FSEDIT statement.

Informats that are specified in an INFORMAT statement take precedence over informats that are defined in the data set itself. Informats that are assigned in an INFORMAT statement remain in effect only for the duration of the procedure; the INFORMAT statement does not affect any informat assignments that are stored in the data set.

When using INFORMAT statements with the FSEDIT procedure, you should make sure the informats that you assign to variables are compatible with the formats for those variables. Otherwise, you will complicate the process of editing the values of the variables. For example, suppose the data set that is displayed by the FSEDIT procedure contains a variable AMOUNT that is assigned the informat 10.2 and the format DOLLAR10.2. Because of the format, values in the field for the variable AMOUNT are displayed with commas and a leading dollar sign:

```
AMOUNT:   $1,250.00
```

However, if you edit this field value (for example, changing it to $1,150.00) and press ENTER, an error condition occurs. The 10.2 informat does not allow the dollar sign ($)

or comma characters in entered values. An appropriate informat for this variable is COMMA., which does allow these characters.

Refer to *SAS Language Reference: Dictionary* for more detailed information about informats and on the INFORMAT statement. See the description of the FORMAT procedure in *Base SAS Procedures Guide* for information about defining your own informats.

LABEL Statement

Associates descriptive labels with variables in the input data set.

Restriction: You must specify the LABEL option in the PROC FSEDIT statement in order for fields in the FSEDIT window to be identified with labels rather than with variable names. Thus, the LABEL statement is useful only in conjunction with the LABEL option.

Reminder: Labels that you specify in the LABEL statement are ignored if you specify an existing SCREEN entry in the SCREEN= option of the PROC FSEDIT statement.

LABEL *variable='label'* <*... variable-n='label-n'*>;

Arguments

At least one pair of the following arguments is required:

variable
is the name of a variable from the input data set.

label
is a string up to 256 characters long.

Using the LABEL Statement

By default, the FSEDIT procedure identifies each field in the FSEDIT window with the corresponding variable name. Labels can be longer and more informative than variable names.

Variable labels that are specified in a LABEL statement take precedence over labels that are stored in the data set itself. The LABEL statement does not affect any variable labels that are stored in the data set.

VAR Statement

Selects which variables to display and what order to display then in.

Reminder: The VAR statement is ignored if an existing SCREEN entry is specified in the SCREEN= option of the PROC FSEDIT statement.

Restriction: The VAR statement causes an error if it is used in conjunction with a PROC FSEDIT statement that includes the NEW= option.

Tip: When more than one variable name is supplied, any valid form of variable list can be used. See *SAS Language Reference: Concepts* for details about variable lists.

VAR *variable* <*... variable-n*>;

Argument

variable is the name of a variable from the input data set.

Using the VAR Statement

By default, the FSEDIT procedure displays all of the variables in the data set and arranges the variables in the order in which they appear in the data set. You can use the VAR statement to control which variables appear in the FSEDIT window and in what order they appear. The maximum number of variable fields that are visible at any given time depends on the width and height of the FSEDIT window.

During an FSEDIT session, you can modify the display to add or remove variable fields.

WHERE Statement

Defines criteria that observations must meet in order to be displayed for editing.

Restriction: The WHERE statement causes an error if it is used in conjunction with a PROC FSEDIT statement that includes the NEW= option.

WHERE *expression*;

Argument

expression is any valid WHERE expression that includes one or more of the variables in the input data set. Refer to the description of the WHERE statement in *SAS Language Reference: Dictionary* for details about the operators and operands that are valid in WHERE expressions.

Using the WHERE Statement

By default, the FSEDIT procedure displays all of the observations in the data set. The WHERE statement is useful when you want to process only a subset of the observations in a SAS data set. For example, to process only observations for which the value of the variable YEAR is less than 5, follow the PROC FSEDIT statement with this statement:

```
where year<5;
```

The FSEDIT procedure displays only observations that meet the specified condition(s). Observations that do not satisfy the condition(s) are not shown and cannot be edited. If no observations meet the specified condition(s), a blank observation is displayed that has observation number 0. In this case, a warning message is also displayed in the window's message line.

Conditions that are imposed by a WHERE statement (or, equivalently, by a WHERE= data set option) are called permanent WHERE clauses because they remain in effect for

the duration of the FSEDIT session and cannot be canceled or modified while the procedure is active.

When you use a WHERE statement in conjunction with the PROC FSEDIT statement, the behavior of the FSEDIT procedure is affected in the following ways:

- The word *Subset* appears in parentheses following the FSEDIT window title to indicate that a permanent WHERE clause is in effect.
- Observation numbers might not be sequential because some observations might be excluded.
- You cannot scroll directly to a particular observation by typing the corresponding observation number on the command line.

FSEDIT Command Syntax

Tip: The FSEDIT command provides an easy way to open an FSEDIT window from any SAS System command line.

FSEDIT <? | *data-set* <*screen-name*>>

FSEDIT Command

Initiates an FSEDIT session.

FSEDIT <? | *data-set* <*screen-name*>>

Arguments

?
opens a selection window from which you can choose the data set to be processed by the FSEDIT procedure. The selection list in the window includes all data sets in all SAS data libraries that have been identified in the current SAS session (all data libraries that have defined librefs).

To select a data set, position the cursor on the desired data set name and press ENTER.

data-set
specifies the data set to be processed by the FSEDIT procedure. The general form of the argument is

<*libref.*>*data-set-name* <(*data-set-options*)>

If you omit the libref, then the default library, WORK, is assumed.

If you specify a data set that does not exist, a selection window is opened showing all available data sets. An error message in the selection window indicates that the specified data set does not exist.

If you omit this argument altogether and do not specify **?** for a selection window, then the most recently created data set (the data set that is identified in the _LAST_= system option) is selected. If no data set has previously been created in the current SAS session, a selection window is opened showing all available data sets. An error message in the selection window indicates that no default data set is available.

You can add a list of data set options following the data set name. The list must be enclosed in parentheses. The FIRSTOBS= and OBS= options are ignored; all other data set options are valid. Refer to *SAS Language Reference: Dictionary* for a list and descriptions of data set options.

screen-name
specifies a SCREEN entry that contains custom field attributes and window characteristics for the FSEDIT session. The general form of the argument is

<*libref.*>*catalog-name*<*.entry-name*<.SCREEN>>

You can specify a one-, two-, three-, or four-level name:

- If a one-level name is specified, it is treated as a catalog name in the default library, WORK. If the specified catalog does not already exist in the WORK library, it is created. The procedure attempts to load the default SCREEN entry FSEDIT.SCREEN from the catalog when the FSEDIT session is initiated. If the default entry does not exist, it is created when a MODIFY command is used during the FSEDIT session.

  Remember that all catalogs in the WORK library are deleted when you terminate your SAS session.

- If a two-level name is specified, it is treated as *libref.catalog-name*. If the specified catalog does not already exist in the specified library, it is created. The procedure attempts to load the default SCREEN entry FSEDIT.SCREEN from the catalog when the FSEDIT session is initiated. If the default entry does not exist, it is created when a MODIFY command is used during the FSEDIT session.

- If a three-level name is specified, it is treated as *libref.catalog-name.entry-name*. The entry type is assumed to be SCREEN. The procedure attempts to load the specified SCREEN entry when the FSEDIT session is initiated.

  If the specified SCREEN entry is not found, the FSEDIT session is initiated with the default display format. However, if the MODIFY command is used during the FSEDIT session, a SCREEN entry that has the specified name is created to store customization information.

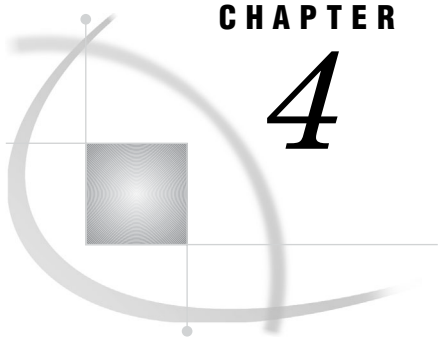
- If a four-level name is specified, the fourth level (entry type) should be SCREEN. Any other value for the type is ignored; SCREEN is used instead. The procedure attempts to load the specified SCREEN entry when the FSEDIT session is initiated.

  If the specified SCREEN entry is not found, the FSEDIT session is initiated with the default display format. However, if the MODIFY command is used during the FSEDIT session, a SCREEN entry that has the specified name is created to store customization information.

Using the FSEDIT Command

The FSEDIT command can be issued in any SAS System window. When you end an FSEDIT session that was initiated with the FSEDIT command, control returns to the window from which the command was issued.

The FSEDIT command does not permit you to specify procedure options such as KEYS= and ADD. You must use a PROC FSEDIT statement rather than the FSEDIT command to initiate the procedure with these options. You must use the PROC FSEDIT statement to produce letters or other documents from the FSEDIT session because the FSEDIT command does not provide a substitute for the statement's LETTER= option. You must also use the PROC FSEDIT statement if you want to modify the procedure's behavior using the FORMAT, INFORMAT, LABEL, VAR, or WHERE statements.



CHAPTER

4

FSEDIT Procedure Windows

Overview

You can use the FSEDIT procedure to perform a variety of tasks. Each task has its own associated window, as shown in the following table:

Task	***Window***
Viewing and editing observations	FSEDIT
Creating a new SAS data set	FSEDIT NEW
Customizing the FSEDIT session	FSEDIT Menu
• redesigning the display	FSEDIT Modify
• defining special fields	FSEDIT Names
• identifying field locations	FSEDIT Identify
• writing an SCL program	FSEDIT Program
• assigning field attributes	FSEDIT Attribute
• setting session parameters	FSEDIT Parms

The following sections explain

- how these tasks are performed with the FSEDIT procedure
- how the associated windows are used
- which commands are valid in each window.

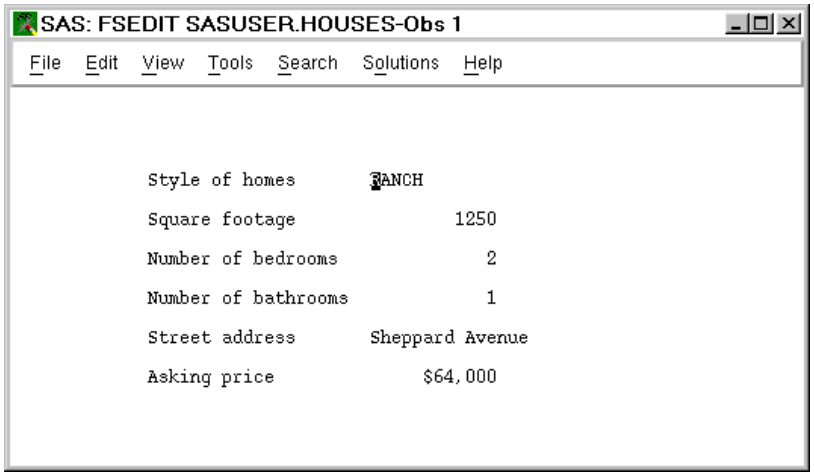
Note: Most of the features that are described in the following sections for the FSEDIT procedure are also available for the FSBROWSE procedure. However, the FSEDIT functions that relate to creating new data sets and to adding, editing, or deleting observations in existing data sets are not applicable to the FSBROWSE procedure. △

Viewing and Editing Observations

In the FSEDIT procedure, observations are viewed and edited in the FSEDIT window. By default, this window is opened when you begin an FSEDIT session. If you use the procedure to create a new data set, the FSEDIT window is opened after the structure of the new data set has been defined in the FSEDIT NEW window.

Display 4.1 on page 25 shows the features of a typical FSEDIT window. The window includes fields that contain the values of variables in the data set, as well as labels that identify the fields.

Display 4.1 Typical FSEDIT Window



By default, the window's title bar includes both the name of the current data set and the current observation number. No observation number is displayed when the engine that is being used to read the data set does not support access by observation number. For example, observation numbers are not displayed when the data set is compressed.

Unless you use a WHERE statement in conjunction with the PROC FSEDIT statement, all observations in the data set are available for editing. The FSEDIT procedure ignores the FIRSTOBS= and OBS= system options.

CAUTION:

The FSEDIT procedure edits a data set in place. The FSEDIT procedure does not leave an unedited copy of the original. If you need to preserve a copy of the original data, be sure to make a copy of the data set before you begin editing. △

How the Control Level Affects Editing

The editing behavior of the FSEDIT procedure depends on which control level is selected when the data set is opened. The *control level* is the degree to which the procedure can restrict access to the data set.

The FSEDIT procedure supports two levels of control:

record — locks only the observation that is currently being edited. With this control level, you can open multiple FSEDIT windows for browsing or editing the same data set. Using SAS/SHARE software, other users can edit the same data set simultaneously.

member — locks the entire data set. No other window or user can open the data set while this control level is in effect.

By default, the FSEDIT procedure selects record-level control when it opens a SAS data set. You can specify the control level with the UPDATE command in the FSEDIT window, or by using the CNTLLEV= data set option with the data set name in the PROC FSEDIT statement or in the FSEDIT command. See "FSEDIT Window Commands" on page 26 for details about the UPDATE command. The CNTLLEV= data set option is described in *SAS Language Reference: Dictionary*.

Scrolling

When the FSEDIT window is opened, an initial observation is displayed for editing. Scroll forward or backward to view other observations.

If an observation contains more variables than can be displayed in the FSEDIT window at one time, the information in each observation is divided into discrete units called *screens*. Each screen contains as many variable fields as will fit in the FSEDIT window. Scroll right or left to move among the screens to view the additional variables.

Note: The FSEDIT window supports a maximum of 100 screens per observation. If you attempt to open a data set that has an extremely large number of variables, such that more than 100 screens would be required to accommodate all of the variables, then only variables that fit within 100 screens are displayed. You can use the VAR statement in conjunction with the PROC FSEDIT statement to restrict the number of variables that are displayed in the FSEDIT window. △

Adding Observations

There are two ways to add observations to the data set:

- Create a new blank observation and enter variable values.
- Duplicate an existing observation and edit the variable values.

The new observation is not actually added to the data set until you move to another observation, save the data set, or end the procedure. You can cancel the observation before it is added to the data set.

Entering and Editing Variable Values

To enter a value for a variable, simply type the value in the entry field (usually indicated by underscores) that follows the variable name or label. To edit a value, type the new value over the old value.

When an observation is displayed for editing, you can enter values only on the command line and in entry fields. All other areas of the window are protected.

FSEDIT Window Commands

In addition to the global commands that are discussed in Chapter 9, "SAS/FSP Software Global Commands," on page 143, you can use the following commands while editing observations:

Scrolling

n

=n

=variable

BACKWARD

BOTTOM

FORWARD

LEFT

RIGHT

TOP

Searching

FIND *search-criterion <... search-criterion-n>*

FIND@ *search-criterion <... search-criterion-n>*

LOCATE | LOC *search-value*

LOCATE: | LOC: *search-string*

NAME <*variable*>

RFIND

SEARCH *search-string*

SEARCH@ *search-string <... search-string-n>*

STRING *variable <... variable-n>*

Editing Observations

ADD

CANCEL

CURSOR

DELETE

DUP

OVERRIDE

UPDATE <RECORD | MEMBER>

Saving Data

AUTOSAVE <*n*>

END

SAVE

Creating Letters and Reports

(These commands are valid only if the LETTER= option is used in the PROC FSEDIT statement.)

EDIT *letter-name*

LETTER

SEND *letter-name*

Other

KEYS

MODIFY <*password*>

REREAD

WHERE <<ALSO> *expression*> | <UNDO | CLEAR>

Command Descriptions

Here are descriptions of the FSEDIT window commands:

n
displays the specified observation. If the *n* value is greater than the number of observations in the data set, the last observation in the data set is displayed.

This command is not valid when the engine that is being used to read the data set does not support access by observation number or when a permanent or temporary WHERE clause is in effect.

=*n*
displays the specified screen of the current observation in a multiscreen application. If the =*n* value is greater than the number of screens in the application, the highest-numbered screen of the current observation is displayed.

This command has no effect if the FSEDIT window does not use multiple screens.

=*variable*
positions the cursor on the entry field for the specified variable.

This command is particularly useful in multiscreen applications and in custom displays.

ADD
creates a new blank observation for the data set and displays it so that you can enter values.

The new observation is not actually added to the data set until you scroll to another observation, issue a SAVE command, or end the FSEDIT session. You can use the CANCEL or DELETE commands to cancel the new observation before it is added to the data set.

AUTOSAVE <*n*>
specifies how frequently the procedure automatically saves the data set. The *n* value determines the number of observations that must be modified (changed, added, or deleted) before an automatic save is performed. By default, the FSEDIT procedure saves the data set automatically whenever 25 observations have been modified since the last save.

To check the current value of the AUTOSAVE parameter, issue the AUTOSAVE command without specifying an *n* value.

When creating a FSEDIT application, you can change the default AUTOSAVE value by changing the **`Autosave value`** field in the FSEDIT Parms window.

Regardless of the AUTOSAVE value, you can save the data set at any time by using the SAVE command.

BACKWARD
displays the previous observation.

BOTTOM
displays the last observation in the data set.

Note: Some engines do not support the BOTTOM command. △

CANCEL
cancels all changes that have been made to the current observation. You can cancel changes only while the observation is displayed. Once you scroll to another observation or issue a SAVE command, the changes cannot be canceled.

CURSOR
selects the position on the display (usually in a variable field) where the cursor is positioned each time an observation is displayed. To specify the position, type CURSOR on the command line, move the cursor to the desired position, and press ENTER.

DELETE
marks the displayed observation for deletion. After you move to another observation, you cannot return to a deleted one.

Depending on which engine is used, deleted observations may not be physically removed from the data set, even though they are no longer accessible. To remove deleted observations, use a DATA step or any other process, such as the SORT procedure, that re-creates the data set.

Note: Some engines, such as the V5 engine, do not support deleting observations. In this case, the DELETE command merely resets all variables in the observation to missing values. △

DUP
creates a duplicate of the displayed observation and displays the newly created observation for editing. Duplicating an observation is useful when you are adding an observation whose values are similar to those of an existing observation.

The duplicate observation is not actually added to the data set until you scroll to another observation, issue a SAVE command, or end the FSEDIT procedure. You can use the CANCEL or DELETE commands to cancel the new observation before it is added to the data set.

EDIT *letter-name*
initiates the FSLETTER procedure and displays the specified document for editing in the FSLETTER window. The *letter-name* value is the one-level name of a LETTER entry in the SAS catalog that is specified in the LETTER= option of the PROC FSEDIT statement that initiates the FSEDIT session. (The EDIT command is valid only when the LETTER= option is used in the PROC FSEDIT statement.) If the LETTER entry does not already exist, it is created.

If you use the SEND command in the FSLETTER window, fields in the document are filled with the values of corresponding variables from the current FSEDIT observation during the FSLETTER send step. When you end the FSLETTER session, the FSEDIT session resumes.

For more information about creating letters and other documents, refer to Chapter 5, "The FSLETTER Procedure," on page 63. See also the LETTER and SEND commands.

END
saves the data set, closes the FSEDIT window, and ends the FSEDIT session.

FIND *search-criterion ... search-criterion-n*
locates and displays the next observation that meets the specified criteria. The general form of the *search-criterion* value is

variable-name comparison-operator search-value

where

- *variable-name* is the name of a variable in the data set. Computed variables cannot be used as search variables.

- *comparison-operator* is one of the following:

=	^= *or* ¬=	>	>=	<	<=
EQ	NE	GT	GE	LT	LE

- *search-value* is a valid value for the variable.

The following restrictions apply to the *search-value* value:

- Character values must be enclosed in quotes if they contain embedded blanks, special characters, or leading numbers.
- Character values must match the case of the variable values, unless the CAPS attribute is assigned to the variable field that is being searched. For example, the following command will not locate observations in which the value of the CITY variable is stored as **Raleigh**:

```
find city=raleigh
```

  You must instead use the following command:

```
find city=Raleigh
```

  When the CAPS attribute is assigned to the variable field that is being searched, the value is converted to uppercase for purposes of the search, regardless of the case in which it is entered.
- Numeric values can be entered either using the standard notation for numeric constants (regardless of which format or informat is associated with the variable) or using the informat that is associated with the variable. If the informatted value contains special characters, the search value must be enclosed in quotes. For example, if a variable named COST has the informat COMMA8.2 and the format DOLLAR10.2, you can specify either of the following to locate an observation in which the **COST** field value is displayed as $1,234.50:

```
find cost=1234.50
find cost='1,234.50'
```

  The FIND command searches informatted values of the specified variable. If the variable has a decimal value, then you must specify at least the decimal point in the search value. For example, if a variable that is named COST has the informat 5.2 and the value 6.00, then searching for the value 6 would not find a match, but searching for the value 6. would.
- Date values must be enclosed in quotes.

The command plus the string cannot be longer than 256 characters. Also, you cannot specify more than 20 find variables.

If a list of criteria is specified, all those criteria must be met in order for an observation to be selected. For example, the following command locates only observations for which the YRS variable contains the value 3 and the STATE variable contains NC:

```
find yrs=3 state=NC
```

The command will not locate observations that meet only one of the criteria. Use the FIND@ command to locate observations that meet some but not all of the conditions in the list.

After you issue a FIND command, you can use the RFIND command to repeat the search for the next matching observation.

You can interrupt a FIND operation that is in progress. This feature is useful when you want to halt a search request while editing or browsing a large data set. To halt an active FIND operation, press the interrupt key or key combination for your system. The FSEDIT procedure halts the operation and displays the following message:

```
NOTE: Search was discontinued due to user break request.
```

To resume the search, issue an RFIND command.

Note: The key or key combination you use to interrupt an active process depends on your host operating system and terminal device. For example, some systems have a key that is labeled BREAK or ATTENTION (or ATTN). Other systems use a combination of the CTRL key and another key. Refer to your host documentation if you are unfamiliar with the interrupt key for your operating system and terminal device. △

See also the FIND@ and RFIND commands.

FIND@ *search-criterion <... search-criterion-n>*
locates and displays the next observation that meets at least one criterion in a list of criteria. For example, use the following command to locate an observation that has either a value greater than 1 for the variable YRS or the value **NC** for the variable STATE:

```
find@ yrs>1 state=NC
```

See the discussion of the FIND command for an explanation of the format for *search-criterion* values.

After you issue a FIND@ command, you can use the RFIND command to repeat the search for the next matching observation.

FORWARD
displays the next observation.

KEYS
opens the KEYS window for browsing and editing function key definitions for the FSEDIT window. Function key definitions are stored in catalog entries of type KEYS.

The default key definitions for the FSEDIT window are stored in the FSEDIT.KEYS entry in the SASHELP.FSP catalog. If you are using this default set of key definitions when you issue the KEYS command and you change any key definitions in the KEYS window, a new copy of the FSEDIT.KEYS entry is created in your personal PROFILE catalog (SASUSER.PROFILE, or WORK.PROFILE if the SASUSER library is not allocated). The changes that you make are recorded in your personal copy of the KEYS entry.

Note: The FSEDIT and FSBROWSE procedures use the same default KEYS entry. Changes that you make in the FSEDIT window also affect the default key definitions for the FSBROWSE window. △

If your FSEDIT session does not use an existing SCREEN entry, you can specify a KEYS entry for your FSEDIT session by using the KEYS= option with the PROC FSEDIT statement. If you do use an existing SCREEN entry, the KEYS entry name that is recorded in the SCREEN entry is used. If you issue a KEYS command when a KEYS entry has been specified, the FSEDIT procedure looks for that entry first in the catalog that contains the SCREEN entry (if a SCREEN entry is used), then in your personal PROFILE catalog. The first KEYS entry found that has the specified name is opened for editing.

If the specified KEYS entry is not found in either catalog, then all function key definitions are blank when the KEYS window is opened. If you then enter key definitions, the specified entry is created when the KEYS window is closed. The new entry is created in the catalog that contains the current SCREEN entry if a SCREEN entry is used; otherwise, it is created in your personal PROFILE catalog.

LEFT
scrolls to the previous screen of the current observation. This command is valid only in multiscreen applications.

LETTER
initiates the FSLETTER procedure and opens the FSLETTER DIRECTORY window to display the directory of the SAS catalog that was specified in the LETTER= option. This command is valid only if you specify the LETTER= option in the PROC FSEDIT statement that initiates the FSEDIT session.

From the FSLETTER DIRECTORY window, you can create new documents, or you can select existing documents for editing or printing. If you issue a SEND command in the FSLETTER window, fields in the document are filled with the values of the corresponding variables from the current FSEDIT observation. When you end the FSLETTER session, the FSEDIT session resumes.

LOCATE *search-value*
LOC *search-value*
locates and displays the next observation that contains a variable value that exactly matches the specified numeric or character value. The FSEDIT procedure searches for the matching value in the variable field that was identified in the most recent NAME command.

The following restrictions apply to the *search-value* value:

- Character values must be enclosed in quotes if they contain embedded blanks or special characters.
- Character values must match the case of the variable values, unless the CAPS attribute is assigned to the variable field that is being searched. For example, the following command will not locate observations in which the CITY variable value is stored as **Raleigh**:

```
locate raleigh
```

  You must instead use the following command:

```
locate Raleigh
```

  When the CAPS attribute is assigned to the variable field that is being searched, the value is converted to uppercase for purposes of the search, regardless of the case in which it is entered.
- Numeric values must be entered using the standard notation for numeric constants, regardless of the format or informat that is associated with the variable. For example, if a variable named COST has the informat COMMA8.2 and the format DOLLAR10.2, you must specify the following command to locate an observation in which the **COST** field value is displayed as $573.04:

```
locate 573.04
```

- Date values must be enclosed in quotes.
- The command plus the string cannot exceed 256 characters.

The LOCATE command finds only observations for which the specified search value exactly matches the variable value. Use the LOCATE: or SEARCH command to find partial matches.

After you issue a LOCATE command, you can use the RFIND command to repeat the search for the next observation that contains the specified value.

You can interrupt a LOCATE operation that is in progress. This feature is useful when you want to halt a search request while editing or browsing a large data set. To halt an active LOCATE operation, press the interrupt key or key combination for your system. The FSEDIT procedure halts the operation and displays the following message:

```
NOTE: Search was discontinued due to user break request.
```

To resume the search, issue an RFIND command.

Note: The key or key combination you use to interrupt an active process depends on your host operating system and terminal device. For example, some systems have a key that is labeled BREAK or ATTENTION (or ATTN). Other systems use a combination of the CTRL key and another key. Refer to your host documentation if you are unfamiliar with the interrupt key for your operating system and terminal device. △

See also the LOCATE:, NAME, and RFIND commands.

LOCATE: *search-string*
LOC: *search-string*

locates and displays the next observation that contains a variable value for which the beginning characters match the specified character value. The FSEDIT procedure searches for the matching value in the variable field that was specified in the most recent NAME command. See the description of the LOCATE command for a list of restrictions on the search value.

Note: For numeric variables, the LOCATE: command finds only exact matches (like the LOCATE command). You cannot search for partial matches in numeric variables. △

The LOCATE: command finds only observations for which the specified search value matches the beginning characters of the variable value. For example, the following command finds occurrences of both Burlington and Burnsville:

```
locate: Bur
```

Use the SEARCH command to find matches anywhere in the variable value rather than just at the beginning.

After you issue a LOCATE: command, you can use the RFIND command to repeat the search for the next observation that contains the specified value.

See also the LOCATE, NAME, and RFIND commands.

MODIFY <*password*>

opens the FSEDIT Menu window, from which you can customize the appearance and behavior of the FSEDIT environment.

If the application you are using is password-protected, you must specify the assigned password with the MODIFY command before you can modify the SCREEN entry.

NAME <*variable*>

specifies the data set variable that will be searched by subsequent LOCATE or LOCATE: commands. (Computed variables cannot be used as search variables.) Issue the NAME command alone to display the current NAME variable.

For example, to find observations that contain particular values of a variable named DISTRICT, issue this command:

```
name district
```

Then specify the desired district value in a LOCATE command to find observations that belong to a specific district.

In an application, you can specify a default search variable in the FSEDIT Parms window.

See also the LOCATE and LOCATE: commands.

OVERRIDE
cancels all outstanding error conditions, permitting you to exit an observation even though you have entered values that are outside of the acceptable range or have left some required fields empty. Values that are flagged as outside the acceptable range are recorded in the data set as entered. Values for fields in which nothing was entered are recorded as missing values.

Note: When you are using an FSEDIT application, the OVERRIDE command is valid only if the application allows overriding. Applications developers can block the overriding of error conditions that are caused by blank required fields, by values outside the acceptable range, or both. △

REREAD
updates the current observation with the saved variable values from the data set.

RFIND
repeats the most recent FIND, FIND@, LOCATE, LOCATE:, SEARCH, or SEARCH@ command.

RIGHT
scrolls to the next screen of the current observation. This command is valid only in multiscreen applications.

SAVE
saves the SAS data set that you are editing without ending the FSEDIT session. You can issue a SAVE command at any time while you are editing observations.

See also the AUTOSAVE command.

SEARCH <*search-string*>
locates and displays the next observation that contains a variable value that includes the specified character value. (The SEARCH command is valid only for character variables.) The FSEDIT procedure searches for the value in the variable fields that were identified in the most recent STRING command.

The following restrictions are applicable to the *search-string* value:

- Values must be enclosed in quotes if they contain embedded blanks or special characters.
- Values must match the case of the variable values, unless the CAPS attribute is assigned to the variable fields that are being searched. For example, the following command will not locate observations in which the CITY variable value is stored as **`Raleigh`**:

  ```
  search raleigh
  ```

  You must instead use the following form of the command:

  ```
  search Raleigh
  ```

  When the CAPS attribute is assigned to the variable field that is being searched, the value is converted to uppercase for purposes of the search, regardless of the case in which it is entered.
- The command plus the string cannot exceed 256 characters.

If a list of values is specified, all of the strings must occur in an observation in order for it to be located. For example, the following command locates only observations for which the specified variables include both the strings **Smith** and **NC**:

```
search Smith NC
```

The strings can occur in two different variable values (if more than one variable is named in the STRING command) or both in the same variable value.

To find observations that contain some but not necessarily all of the values in the list, use the SEARCH@ command.

After you issue a SEARCH command, you can use the RFIND command to repeat the search for the next observation that contains the specified value.

You can interrupt a SEARCH operation that is in progress. This feature is useful when you want to halt a search request while editing or browsing a large data set. To halt an active SEARCH operation, press the interrupt key or key combination for your system. The FSEDIT procedure halts the operation and displays the following message:

```
NOTE: Search was discontinued due to user break request.
```

To resume the search, issue an RFIND command.

Note: The key or key combination you use to interrupt an active process depends on your host operating system and terminal device. For example, some systems have a key labeled BREAK or ATTENTION (or ATTN). Other systems use a combination of the CTRL key and another key. Refer to your host documentation if you are unfamiliar with the interrupt key for your operating system and terminal device. △

See also the SEARCH@, STRING, and RFIND commands.

SEARCH@ *search-string* <... *search-string-n*>

locates and displays the next observation that contains variable values that include one or more of the specified character values. (The SEARCH@ command is valid only for character values.) The FSEDIT procedure searches for the values in the variables that were identified in the most recent STRING command. See the description of the SEARCH command for restrictions that apply to the *character-string* value.

For example, the following command displays the next observation that contains either **Cary**, **Raleigh**, or **Chapel Hill** in one of the variables identified in the STRING command:

```
search@ Cary Raleigh 'Chapel Hill'
```

After you issue a SEARCH@ command, you can use the RFIND command to repeat the search for the next observation that contains the specified value.

See also the SEARCH, STRING, and RFIND commands.

SEND *letter-name*

initiates the FSLETTER procedure in its send step, displays the specified document, and fills any entry fields in the document with corresponding variable values from the current observation. The SEND command is valid only when the LETTER= option is used in the PROC FSEDIT statement that initiates the FSEDIT session.

The *letter-name* value is the one-level name of an existing LETTER entry in the SAS catalog that is specified in the LETTER= option of the PROC FSEDIT statement. An error message is printed if the specified LETTER entry does not exist.

Use the END command to enter the second stage of the send step, or use the CANCEL command to cancel the FSLETTER session. When you end the FSLETTER session, the FSEDIT session resumes.

The SEND command provides a method for producing one copy of a document for one observation. See Chapter 5, "The FSLETTER Procedure," on page 63 for more details.

STRING <*variable* <*... variable-n*>>
identifies the data set variable or variables that will be searched by subsequent SEARCH and SEARCH@ commands. The variables that are specified with the command must be character variables in the data set. Computed variables cannot be used as search variables.

For example, the following command causes the next SEARCH or SEARCH@ command to search the two specified variables:

```
string address1 address2
```

If you forget which variables are currently identified, issue the STRING command with no following values to display the current variables on the window's message line.

In a custom application, you can specify default search variables in the FSEDIT Parms window.

TOP
displays the first observation in the data set.

UPDATE <RECORD | MEMBER>
changes the control level of an FSEDIT window.

The UPDATE command fails if the specified control level would cause a locking conflict. For example, you cannot specify UPDATE MEMBER if the same data set is open with a control level of RECORD in another FSEDIT session.

WHERE <<ALSO> *expression*> | <UNDO | CLEAR>
imposes one or more sets of conditions that observations in the data set must meet in order to be processed. *Expression* is any valid WHERE expression that includes one or more of the variables in the input data set. (Refer to the description of the WHERE statement in *SAS Language Reference: Dictionary* for details about the operators and operands that are valid in WHERE expressions.) Observations that do not satisfy the specified conditions cannot be displayed or edited.

The complete set of conditions that are imposed by a WHERE command is called a temporary WHERE clause. These conditions can be modified or canceled during the FSEDIT session. In contrast, a WHERE statement that is submitted by the PROC FSEDIT statement defines a permanent WHERE clause that cannot be changed or canceled during the FSEDIT session and which is not affected by WHERE commands. See "WHERE Statement" on page 20 for details.

The word *Where* appears in the upper right corner of the window border whenever a temporary WHERE clause is in effect.

The WHERE command has the following forms:

WHERE *expression*
applies the conditions that are specified in *expression* as the new temporary WHERE clause, replacing any clause previously in effect.

WHERE ALSO *expression*
adds the conditions that are specified in *expression* to any existing temporary WHERE clause.

WHERE UNDO
deletes the most recently added set of conditions from the temporary WHERE clause.

WHERE
WHERE CLEAR
cancels the current temporary WHERE clause.

Whenever you change the temporary WHERE clause, the procedure scrolls to the first observation in the data set that meets the specified conditions. When you cancel the temporary WHERE clause, the procedure displays the first observation in the data set.

If you edit values in an observation so that it no longer meets the conditions of the WHERE clause, that observation can still be displayed and be edited. However, a warning message is printed whenever the observation is displayed, indicating that the observation no longer meets the WHERE conditions.

When you use the ADD or DUP commands to add a new observation, you can enter values that do not meet the WHERE conditions. However, once you scroll to another observation, that observation cannot be displayed or edited again while the WHERE clause is in effect.

Creating a New Data Set

You can use the FSEDIT procedure to create a SAS data set. You name the variables and specify their attributes in fields in the FSEDIT NEW window. After you exit the FSEDIT NEW window, the data set is created and the FSEDIT window is opened so that you can enter values in the new data set.

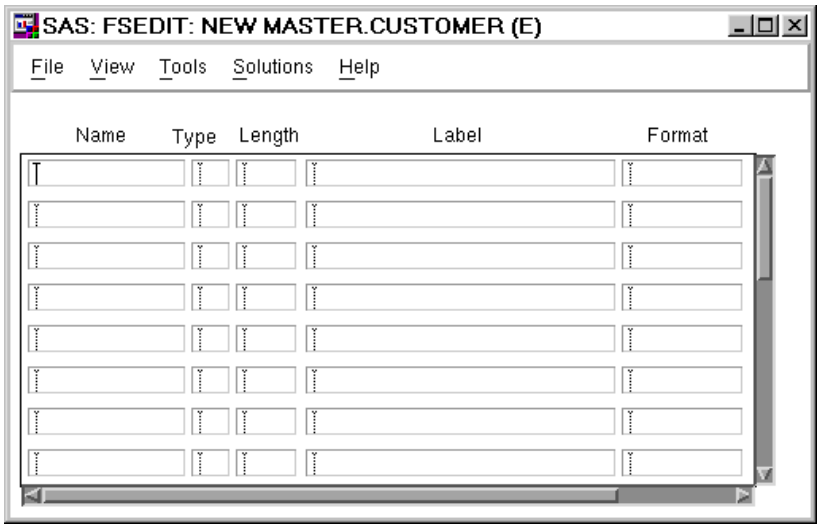
Opening the FSEDIT NEW Window

To open the FSEDIT NEW window, invoke the FSEDIT procedure, using the NEW= option in the PROC FSEDIT statement. For example, to create a data set that is named CUSTOMER in the SAS data library that has the libref MASTER, submit the following statements:

```
proc fsedit new=master.customer;
run;
```

Display 4.2 on page 38 shows the FSEDIT NEW window that is opened when these statements are submitted.

Display 4.2 The FSEDIT NEW Window



Defining Variables

The following rules apply to defining variables in the FSEDIT NEW window:

- You must give each variable a name. The name must follow SAS naming conventions. See *SAS Language Reference: Concepts* for details.
- You can identify the type for each variable. Use **N** for numeric or **$** (or **C**) for character. If you leave the **Type** field blank, the default type is numeric.
- You can specify the length of each variable. If you leave the **Length** field blank, the default length is 8.
- You can assign a label, a format, and an informat for each variable. See *SAS Language Reference: Concepts* for a complete discussion of SAS variable attributes.

If you want to create a data set whose variable names and attributes are identical or similar to those of an existing data set, use the LIKE= option in conjunction with the NEW= option. The LIKE= option initializes the fields of the FSEDIT NEW window with the names and attributes of the variables in the specified data set. You can edit any of the variable names and attributes, and you can define additional variables before creating the data set.

Closing the FSEDIT NEW Window

Use the END command to close the FSEDIT NEW window. This command also creates the data set and opens the FSEDIT window for adding observations to the newly created data set. After you issue the END command, you cannot return to the FSEDIT NEW window to make structural changes to the data set.

FSEDIT NEW Window Commands

In addition to the global commands that are discussed in Chapter 9, “SAS/FSP Software Global Commands,” on page 143, you can use the following commands in the FSEDIT NEW window to scroll through information, to duplicate selected lines, or to exit with the choice of creating a data set or canceling it.

Scrolling

BACKWARD <HALF | PAGE | MAX | *n*>

BOTTOM

FORWARD <HALF | PAGE | MAX | *n*>

LEFT

RIGHT

TOP

Other

CANCEL

END

KEYS

Command Descriptions

Here are descriptions of the FSEDIT New window commands:

BACKWARD <HALF | PAGE | MAX | *n*>
scrolls vertically toward the top of the window. The following scroll amounts can be specified:

HALF	scrolls upward by half the number of lines in the window.
PAGE	scrolls upward by the number of lines in the window.
MAX	scrolls upward until the first line is displayed.
n	scrolls upward by the specified number of lines.

The default scroll amount is HALF.

BOTTOM
scrolls downward until the last line that contains a variable definition is displayed.

CANCEL
closes the FSEDIT NEW window and ends the FSEDIT session. The new data set is not created.

END
closes the FSEDIT NEW window, creates the SAS data set that is defined in the window, and opens an FSEDIT window for adding observations to the newly created data set.

FORWARD <HALF | PAGE | MAX | *n*>
scrolls vertically toward the bottom of the window.

Note: You can scroll forward only if you have filled the last blank variable-definition line that is currently displayed, or if there are more variables to be displayed. △

The following scroll amounts can be specified:

HALF	scrolls downward by half the number of lines in the window.
PAGE	scrolls downward by the number of lines in the window.
MAX	scrolls downward until the last line that contains a variable definition is displayed.

n scrolls downward by the specified number of lines.

The default scroll amount is HALF.

KEYS
opens the KEYS window for browsing and editing function key definitions.

Unlike the other FSEDIT windows, the FSEDIT NEW window uses the default SAS windowing environment KEYS entry rather than the FSEDIT.KEYS entry or the entry that is specified in the KEYS= option if that option is used with the PROC FSEDIT statement.

LEFT
displays the FORMAT column when the INFORMAT column is displayed or vice versa. The RIGHT command has the same effect.

RIGHT
displays the FORMAT column when the INFORMAT column is displayed or vice versa. The LEFT command has the same effect.

TOP
scrolls upward until the first variable-definition line is displayed.

Creating an FSEDIT Application

If you are an applications developer, you can use the FSEDIT procedure as the basis for data entry applications and editing applications. The FSEDIT procedure enables you to customize the application environment to suit the needs of your users. Customization can include

- redesigning the display
- creating special fields
- creating a SAS Component Language program to drive the application
- assigning field attributes to determine how variable values are presented
- setting general parameters that control behavior of the FSEDIT session.

Note: All of the following information about creating FSEDIT applications is equally applicable to creating data presentation applications with the FSBROWSE procedure. △

Storing Customization Information

To create a custom FSEDIT application, you must perform the following steps:

1. Identify the SAS catalog in which information about the customized features is to be stored. Use the SCREEN= option in the PROC FSEDIT statement or the *screen-name* argument in the FSEDIT command to identify the catalog. The procedure can supply a default name for the SCREEN entry, or you can specify a name.
2. Issue the MODIFY command in the FSEDIT window (or use the MODIFY option in a PROC FSEDIT statement) to open the FSEDIT Menu window. From there you can choose from several tasks that are involved in creating a customized application.

Information about the features of an FSEDIT application is stored in a *SCREEN entry*, a SAS catalog entry of type SCREEN. All of the customization information for an application is stored in a single SCREEN entry.

Use the SCREEN= option in the PROC FSEDIT statement or the *screen-name* argument in the FSEDIT command to identify the catalog and, optionally, the entry name. When the FSEDIT procedure is initiated, the procedure looks in the specified catalog for a SCREEN entry. If the catalog does not exist, it is created. If you do not specify an entry name, the procedure looks for an entry that has the default name FSEDIT.SCREEN. If a SCREEN entry that has the designated name is found, a customized FSEDIT session is initiated. If the SCREEN entry does not exist, the FSEDIT session is initiated without customized features. (The SCREEN entry is not created until the MODIFY command is used.)

For example, if you submit the following statements, the procedure looks for an entry named FSEDIT.SCREEN in the MASTER.SCRSUB catalog:

```
proc fsedit data=master.subscrib
     screen=master.scrsub;
run;
```

If the MASTER.SCRSUB catalog does not exist, it is created. If the FSEDIT.SCREEN entry does not exist in the catalog, it is created when the MODIFY command is used for the first time.

If you submit the following statements, the procedure looks for an entry that is named BASIC.SCREEN in the MASTER.SCRSUB catalog:

```
proc fsedit data=master.subscrib
     screen=master.scrsub.basic.screen;
run;
```

If the MASTER.SCRSUB catalog does not exist, it is created. If the BASIC.SCREEN entry does not exist in the catalog, it is created when the MODIFY command is used for the first time. To use the customized application in a future session, users must specify the complete three- or four-level name of the catalog entry. (The fourth level, the entry type, can be omitted because the type for SCREEN entries is always SCREEN.)

Creating or Modifying a SCREEN Entry

The SCREEN entry for an FSEDIT application can hold a variety of information, including

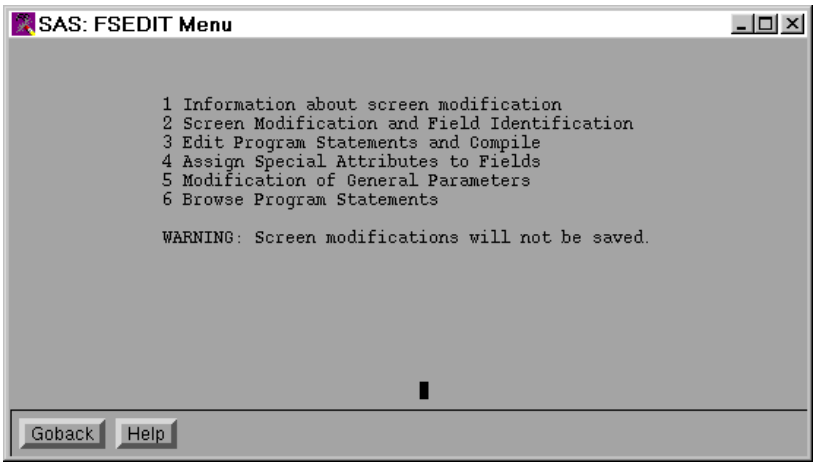
- the customized display format
- a SAS Component Language program
- attribute information for all of the fields
- the general parameters of the FSEDIT session.

Each of these elements is defined or modified in a separate FSEDIT auxiliary window. You must use the FSEDIT Menu window to gain access to any of the auxiliary windows.

Opening the FSEDIT Menu Window

Issue the MODIFY command in the FSEDIT window to open the FSEDIT Menu window. You can also open the FSEDIT Menu window at the beginning of an FSEDIT session, before the FSEDIT window is opened, by using the MODIFY option with the PROC FSEDIT statement. Display 4.3 on page 42 shows the FSEDIT Menu window.

Display 4.3 The FSEDIT Menu Window



To select an option from the main menu, type the option number on the command line and press ENTER. Alternatively, you can move the cursor to the desired item number and press ENTER.

Here are brief explanations of the available options:

Option 1 "Information about Screen Modification" provides information about the tasks that are involved in customizing the FSEDIT application. This option opens a Help window; the effect is the same as using the HELP command in the FSEDIT Menu window.

Option 2 "Screen Modification and Field Identification" enables you to redesign the display, to define special fields, and to identify the variable that is associated with each field. This option opens the FSEDIT Modify window. See “Modifying Screens and Identifying Fields” on page 43 for more information.

Option 3 "Edit Program Statements and Compile" enables you to create and compile a SAS Component Language (SCL) program. This option opens the FSEDIT Program window. See “Editing, Browsing, and Compiling Program Statements” on page 51 for more information.

Option 4 "Assign Special Attributes to Fields" enables you to define or change the attributes of variable fields. This option opens the FSEDIT Attribute window. See “Assigning Special Attributes to Fields” on page 52 for more information.

Option 5 "Modification of General Parameters" enables you to define or change the general parameters of your FSEDIT application. This option opens the FSEDIT Parms window. See “Modifying General Parameters” on page 57 for more information.

Option 6 "Browse Program Statements" enables you to browse an SCL program without compiling it. This option opens the FSEDIT Program window. See “Editing, Browsing, and Compiling Program Statements” on page 51 for more information.

Later sections describe each option and its associated window in greater detail.

Closing the FSEDIT Menu Window

Use the END command to close the FSEDIT Menu window. This command also updates the SCREEN entry and returns you to the FSEDIT window. Any customized

features that you define using the options in the FSEDIT Menu window take effect immediately.

Note: Customization information is not saved after the current FSEDIT session unless you specify the SCREEN= option in the PROC FSEDIT statement or the *screen-name* argument in the FSEDIT command. △

Protecting Your Application

You can protect the integrity of your FSEDIT application by assigning a password to the SCREEN entry. Once the password is assigned, a user of the application must specify it with the MODIFY command in order to change customized features. Others can use your application to edit a SAS data set, but the application itself is protected from unwanted changes to the display, the SCL program, the field attributes, or the FSEDIT general parameter settings.

Passwords are assigned in the **`Modify password`** field of the FSEDIT Parms window. For details, see "Modifying General Parameters" on page 57.

Modifying Screens and Identifying Fields

Select option 2 from the FSEDIT Menu window to create a customized display for your application. Customization is a three-step process:

1. *Modifying the display.* Redesign the display by moving fields, adding fields, or adding descriptive text.
2. *Defining fields.* Specify the attributes of repeated fields, fields for values that are calculated in a SAS Component Language program, or both. This step is necessary only when you add repeated or computed fields, which are described in the next section.
3. *Identifying fields.* Specify the location of the field for each data set variable or computed value.

Step 1: Modifying the Display

The first window that opens when you select option 2 from the FSEDIT Menu window is the FSEDIT Modify window. In this window you design a customized display for your application. Variable fields can be labeled more descriptively, rearranged, and even deleted. You can add comments to help users enter data in the proper format.

The FSEDIT Modify window initially contains the display format for the FSEDIT window (either the default format if a new SCREEN entry is being created, or the previous customized format if an existing SCREEN entry is used.) During this first step, the entire contents of the FSEDIT Modify window are unprotected, so you can type over any area in the display, including the variable names. You can move, delete, or insert any lines in the display. You can move variable fields and add any special comments or instructions that would make entering data easier.

If the modified display format that you create has more lines than the number of rows in the FSEDIT window, a multiscreen application is created. Users must scroll to view the fields and text that do not fit in the first screen. Option 5 in the FSEDIT Menu window enables you to specify the initial height of the FSEDIT window.

Creating Fields

There are three important requirements for variable fields in a customized display:

- Underscore (_) characters are used to define the location and length of fields. The number of underscores you use for a field determines the field width (the number of characters that can be entered in that field).
- Each field must be preceded and followed by at least one blank space, unless the field begins in the leftmost column.
- If a field continues to the next set of underscores, an asterisk (*) must be placed in the last position of a series of underscores, whether the next set is on the same line or on the next line. For example, the following underscores and asterisks define a single field:

```
__* - __* - ____
```

Note: The restriction of using an underscore as the field pad character is applicable only when you are identifying fields to the FSEDIT procedure. This rule does not affect the final appearance of the display. If you want to use a pad character other than the default underscore to mark the location of a variable field, use option 4 from the FSEDIT Menu window to change the PAD attribute for the field. △

The default width of each variable field depends on how the variable is stored in the data set and on whether the variable has an associated output format:*

Variable Type	Default Width
character	the larger of • the width of the variable in the data set • the width of the variable's format or informat (whichever is longer), if one has been assigned.
numeric	either • the width of the variable's format or informat (whichever is longer), if one has been assigned • the default width of 12 (because BEST12. is the default numeric format).

You can modify the default field widths when you create a customized display. For example, many numeric fields do not require the full default width of 12 positions. However, you should ensure that the width of the field is appropriate for the width of the corresponding variable. Otherwise, users of your application may be unable to enter the full range of valid variable values in the fields.

Creating Special Fields

In addition to variable fields, you can create two different types of special fields:

repeated fields

* See *SAS Language Reference: Concepts* for a complete discussion of SAS variable attributes.

repeat the values from other variable fields or computed fields. Repeated fields effectively provide multiple fields for a single variable. Changes that are made in a variable field appear in any repeated fields for that variable, and changes that are made in a repeated field affect the variable field as well as any other repeated fields for that variable.

Repeated fields are useful in multiscreen applications when you want certain fields to appear on more than one screen.

computed fields
display temporary values that are calculated or defined when a SAS Component Language program executes. Although a computed field does not have an associated variable in the input data set, it can be referenced in an SCL program and used for calculations.

These special fields are defined in the same manner as variable fields, with a series of underscores that are preceded and followed either by a blank or by the edge of the window.

FSEDIT Modify Window Commands

When designing a display in the FSEDIT Modify window, you can use all of the SAS/AF global commands and all of the SAS text editor commands.

Note: Because the Modify window uses the SAS text editor, you can use the editor's spell checking feature. To check the spelling of the descriptive text in the window, use the SPELL ALL command. △

Specifying Color and Highlighting

If your terminal or workstation supports color and highlighting, you can change the attributes of the text in your customized display. When your application is used, the color information is ignored if the user's device does not support color. If you have used a color that is not available on the user's device, the procedure substitutes the available color that most closely matches the specified color.

Use the global COLOR TEXT command to change the color and highlighting attributes of the text you enter. For example, the following command changes all of the text you type after the command is issued to high-intensity blue:

```
color text blue h
```

Once you enter a COLOR TEXT command, the specified attributes are used until you change them with another COLOR command. Refer to the description of the COLOR command in the online Help for Base SAS software for additional details.

Note: Some terminals or workstations provide special keys that control text color and highlighting. If your device has such keys, you can use them to set color and highlighting attributes as you enter the text. △

Exiting the FSEDIT Modify Window

Issue the END command to close the FSEDIT Modify window. Before the window is closed, the FSEDIT procedure displays the following question:

```
Did you create any computational or repeated fields (Y or N) ? _
```

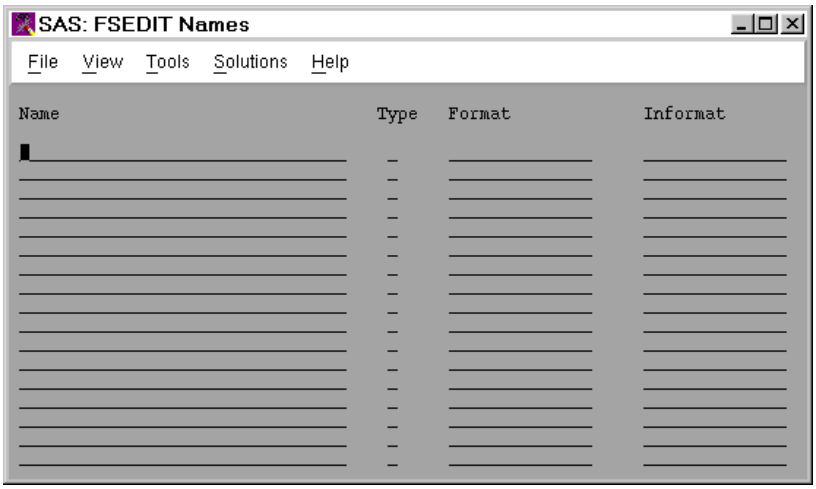
Your response determines whether you go directly to the field identification step or enter the field definition step first.

If you have added any special (computed or repeated) fields, type a **Y** in the space provided. You then enter the field definition step (step 2), where you can define the fields you have added. Otherwise, type an **N** in the space provided. The procedure then takes you directly to the field identification step (step 3).

Step 2: Defining Fields

When you indicate in the FSEDIT Modify window that you have created special fields, the FSEDIT Names window is opened when the FSEDIT Modify window is closed. In the FSEDIT Names window you define the characteristics of special fields. Display 4.4 on page 46 shows the initial FSEDIT Names window display.

Display 4.4 The FSEDIT Names Window



All of the entries in the FSEDIT Names window are initially blank. Special fields that are added during customization are unknown to the FSEDIT procedure until they are defined in the FSEDIT Names window. Special fields are used to hold repeated values or computed values from the program. Do not confuse defining special fields with adding variables to an existing SAS data set.

Defining Special Fields

The rules for defining special fields are similar to the rules for defining SAS variables when you create a new data set:

- Give each special field a name in the **Name** field. The name must follow SAS naming conventions. For repeated fields, use the exact name of the variable that is being repeated.
- Indicate the type of each special field in the **Type** field. Use one of the following characters:

  For computed fields:

  N for numeric fields

  $ (or **C**) for character fields

  For repeated fields:

  R (the field automatically takes the type of the original variable field).

If you do not specify a value in the **Type** field, the default type is **N** (numeric computed).

- Optionally, you can assign a format and an informat to each special field. Repeated fields can have different formats and informats from the original variable field.

Note: For repeated fields, the first occurrence of the field in the display is treated as the original field, the next occurrence is treated as the first repeat, and so on. △

FSEDIT Names Window Commands

In addition to the global commands that are listed in Chapter 9, "SAS/FSP Software Global Commands," on page 143, you can use the following commands in the FSEDIT Names window step to scroll through information, duplicate selected lines, and exit, going directly into the field identification step.

Scrolling

BACKWARD

BOTTOM

FORWARD

TOP

Duplicating

REPEAT

SELECT

Other

END

KEYS

Command Descriptions

Here are descriptions of the FSEDIT Names window commands:

BACKWARD
scrolls toward the top of the window.

BOTTOM
scrolls to the bottom of the window.

END
exits the field definition step and enters the field identification step.

FORWARD
scrolls toward the bottom of the window. You can scroll forward only if you have filled all lines that are currently displayed or if there are more special field names to be displayed.

KEYS
opens the KEYS window for browsing and editing function key definitions. See the description of the KEYS command in "FSEDIT Window Commands" on page 26 for details.

Note: FSEDIT procedure windows share the same KEYS entry. Changes that you make with this command from the FSEDIT Names window will affect the other windows also. △

REPEAT
specifies the target line on which you want a selected line to be repeated. After executing the SELECT command, type REPEAT on the command line, position the cursor on the desired line, and press ENTER. The selected line is then copied to the indicated target line. Unless the field type is **R** (repeated), you receive an error message warning you that the copy is the second occurrence of the field name. To cancel the error, change the name on the copied line.

SELECT
specifies a line whose contents you want to be repeated on another line. Type SELECT on the command line, position the cursor on the line you want to repeat, and press ENTER. The selected line is remembered; any REPEAT command that you issue subsequently will copy the selected line to the desired target line.

TOP
scrolls to the top of the window.

Exiting the FSEDIT Names Window

When you have defined all computational and repeated fields, issue the END command to leave the field definition step. Once all special fields are defined to the procedure, you enter the field identification step, where you identify the locations of all special fields and any variable fields that the FSEDIT procedure has lost track of.

Step 3: Identifying Fields

The FSEDIT Identify window is opened automatically when the FSEDIT Names window is closed, or when the FSEDIT Modify window is closed if no special fields were created during display modification. When the FSEDIT Identify window is opened, the status of each field, whether a data set variable field or a special field, is determined to be one of the following:

identified
The procedure knows the field's location in the display.

unidentified
The procedure does not know the field's location in the display and prompts you to specify the location.

unwanted
The procedure knows that the variable has been omitted from the display and does not prompt you for it. (For example, if you use a VAR statement to select variables to display when you invoke the FSEDIT procedure, all variables in the data set that are not specified in the VAR statement are deemed unwanted by the FSEDIT procedure.)

Before you can exit the field identification step, all fields must be either identified or defined as unwanted. When the FSEDIT procedure knows the location of all variable fields, the following message is displayed:

```
NOTE: All fields are identified.
```

If the FSEDIT procedure does not know the location of a variable field, or if you have added any special fields, you are asked to identify the location of the unidentified fields.

Unidentified Fields

Fields in a customized display can become unidentified in several ways:

- If you perform extensive editing when you modify the display, the FSEDIT procedure may lose track of the location of some variables. Previously identified fields may become unidentified.
- If you add a variable to the data set that is used in the application, you must create a field for the variable in the display for the new variable to be recognized by the FSEDIT procedure. (When you use a customized display, new fields are not automatically added for new data set variables.) A field that you create for the new variable is initially unidentified.
- If you add special fields, they are always initially unidentified. The FSEDIT procedure knows their names but not their locations.

For each unidentified field, you receive a prompt like the following:

```
Please put cursor on field: name and press ENTER ... or UNWANTED
```

To indicate that you are not using a particular variable in the application, issue the UNWANTED command. To identify the location of a variable field or a special field that is being used in the application, position the cursor on any underscore in the appropriate field and press ENTER. Continue to identify fields until a message tells you that all fields are identified.

For example, if you receive the prompt

```
Please put cursor on field: ADDR1 and press ENTER ... or UNWANTED
```

you can do one of the following:

1. Issue the UNWANTED command. (The command can be assigned to a function key.) The FSEDIT procedure then knows that the variable ADDR1 has been purposely excluded from the display, so it does not prompt you again to identify the **ADDR1** field's location.
2. Position the cursor on one of the underscores for the appropriate variable field (**ADDR1** in this example), and press ENTER. The variable ADDR1 changes from unidentified to identified. The FSEDIT procedure then knows the **ADDR1** field's location.

Changing a Field from Unwanted to Identified

If you change your mind about making a variable unwanted, you can use the DEFINE command. Follow DEFINE with the variable name; then position the cursor on the variable field and press ENTER.

If you want to change the status of several variables, you can use the WANTED command. When you issue the WANTED command without specifying any variable names, all unwanted variables become unidentified. The FSEDIT procedure then prompts you to identify the location of all unidentified variable fields.

Notice the difference between these two commands: DEFINE changes a single variable directly from unwanted to identified. WANTED changes one or all variables from unwanted to unidentified. You must then identify the location of each variable's field or define the variable as unwanted again.

FSEDIT Identify Window Commands

In addition to the global commands that are listed in Chapter 9, "SAS/FSP Software Global Commands," on page 143, you can use the following commands in the FSEDIT Identify window:

Identifying Fields

DEFINE *variable*

UNWANTED

WANTED <*variable*>

Scrolling

=*variable*

LEFT

RIGHT

Other

END

KEYS

Command Descriptions

Here are descriptions of the FSEDIT Identify window commands:

=*variable*
locates identified variables. To determine the location of a variable field in the display, type an equal sign on the command line, followed by the variable name, and press ENTER. If the specified variable is an identified variable in the customized display, the cursor then moves to the field for the variable.

DEFINE *variable*
changes the status of a variable from unwanted to identified. Follow DEFINE with the name of the variable, position the cursor on any underscore of the field for that variable, and press ENTER. Remember to use the actual name of the variable instead of a label that you may have assigned to the variable in the customized display.

END
ends the field identification step, closes the FSEDIT Identify window, and returns to the FSEDIT Menu window. This command is not valid until all fields have been either identified or defined as unwanted. If any fields are not currently identified, the FSEDIT procedure prompts you to identify their locations before ending the field identification step.

KEYS
opens the KEYS window for browsing and editing function key definitions. See the description of the KEYS command in "FSEDIT Window Commands" on page 26 for details.

Note: FSEDIT procedure windows share the same KEYS entry. Changes that you make with this command from the FSEDIT Identify window also affect the other windows. △

LEFT
moves to the previous screen of an observation (in multiscreen applications).

RIGHT
moves to the next screen of an observation (in multiscreen applications).

UNWANTED
specifies that a variable field is unwanted and will not be used in this application. To indicate an unwanted variable, issue the UNWANTED command when you are prompted for the location of the variable field.

WANTED <*variable*>
changes the status of a specified variable from unwanted to unidentified. If you do not specify a particular variable, all unwanted variables are changed to unidentified variables. Once a variable becomes unidentified (rather than unwanted), the FSEDIT procedure prompts you to identify its location.

Exiting the FSEDIT Identify Window

You cannot exit the field identification step until you have identified the locations of the fields for all wanted variables and have received the following message:

```
NOTE: All fields are identified.
```

After receiving this message, you can issue the END command to close the FSEDIT Identify window and return to the FSEDIT Menu window.

Editing, Browsing, and Compiling Program Statements

Select option 3 from the FSEDIT Menu window to create, compile, and save a SAS Component Language program for your FSEDIT application. This option opens the FSEDIT Program window, in which you can use all of the SAS text editor commands to enter and edit SCL program statements. Use the END command to compile and save the SCL program in the current SCREEN entry. The END command also closes the FSEDIT Program window and returns you to the FSEDIT Menu window.

Using SAS Component Language

SAS Component Language (SCL) enables you to add power and flexibility to your FSEDIT applications. You can write SCL programs that

- cross-validate values that have been entered in FSEDIT window fields with other variable values in the same SAS data set
- cross-validate values that have been entered in FSEDIT window fields with variable values in other SAS data sets
- manipulate field values based on user input
- manipulate values in other SAS data sets
- manipulate external files
- provide custom messages and help based on user input.

Refer to *SAS Component Language: Reference* for more information about SCL programming.

Browse Program Statements

Select option 6 from the FSEDIT Menu window to browse the current contents of the FSEDIT Program window. When you open the FSEDIT Program window with this

option, all of the SAS text editor browsing commands are valid, but editing the SCL program is prohibited. Use the END command to close the FSEDIT Program window and return to the FSEDIT Menu window.

Assigning Special Attributes to Fields

Select option 4 from the FSEDIT Menu window to define the attributes of each field in the FSEDIT display. This option opens the FSEDIT Attribute window. Use the END command to close the FSEDIT Attribute window and return to the FSEDIT Menu window.

Field Attributes

Field attributes make it easier for users of your application to enter and edit data correctly. See "Attribute Frame Descriptions" on page 53 for more information about each attribute's frame. Each field has the following attributes:

INITIAL
specifies an initial value for the field.

MAXIMUM
specifies the maximum value for the field.

MINIMUM
specifies the minimum value for the field.

REQUIRED
specifies whether a value must be entered in the field when a new observation is added.

CAPS
specifies whether text in the field is converted to uppercase.

FCOLOR
specifies the text color of valid values.

ECOLOR
specifies the text color of invalid values.

FATTR
selects the text highlighting attribute of valid values.

EATTR
selects the text highlighting attribute of invalid values.

PAD
specifies the pad character for the field.

PROTECT
specifies whether the field value can be edited.

JUSTIFY
specifies the text alignment for the field.

NONDISPLAY
specifies whether text in the field is visible.

NOAUTOSKIP
specifies the cursor behavior for the field.

NOAUTOBLANK
specifies how values that are entered in numeric fields are processed.

FSEDIT Attribute Window Frames

The FSEDIT Attribute window is divided into a series of frames, one for each field attribute. Each frame of the FSEDIT Attribute window defines the status of a particular attribute for all of the fields in the customized display. Each frame uses the customized display format that was created for the application.

Scrolling in the FSEDIT Attribute Window

Field attribute frames are stored in the order shown in "Field Attributes" on page 52. You can move from one field attribute frame to another by using the BACKWARD and FORWARD commands. You can also display the frame for a particular attribute by typing its name on the command line and pressing ENTER.

For multiscreen applications, each field attribute frame is also divided into screens. Use the LEFT and RIGHT commands to display fields on successive screens. Use the END command to close the FSEDIT Attribute window and return to the FSEDIT Menu window.

Attribute Frame Descriptions

Here are descriptions of the attribute frames:

INITIAL
assigns initial values to fields. The values that you enter in the fields of this frame are displayed instead of pad characters in the corresponding fields for all new observations that you add to the data set. Initial values that are assigned in this frame do not affect existing values in the data set.

MAXIMUM
assigns the maximum values that can be entered in fields. If a user enters a data value that is greater than the maximum value for that field, an error condition occurs.

This attribute is valid for character fields as well as for numeric fields. For character fields, the "greater than" comparison is based on the operating system's character collating sequence.

By default, users of your application can use the OVERRIDE command to override the error condition that is caused by entering a value greater than the specified maximum. This allows the value to be stored in the data set. You can prevent this by indicating in the **`Override on errors`** field of the FSEDIT Parms window that overriding is not allowed. See "Modifying General Parameters" on page 57 for details.

MINIMUM
assigns the minimum values that can be entered in fields. If a user enters a data value that is less than the minimum value for that field, an error condition occurs.

This attribute is valid for character fields as well as for numeric fields. For character fields, the "less than" comparison is based on the operating system's character collating sequence.

By default, users of your application can use the OVERRIDE command to override the error condition that is caused by entering a value less than the specified minimum. This allows the value to be stored in the data set. You can

prevent this by indicating in the **Override on errors** field of the FSEDIT Parms window that overriding is not allowed. See "Modifying General Parameters" on page 57 for details.

REQUIRED
specifies required fields. When the FSEDIT application is used to add a new observation to the data set, values must be entered in all required fields before the user can leave the observation. A blank or missing value is not considered a valid value unless, in the case of numeric variables, it is a special missing value.

Type an **R** in the first position of a field to indicate a required field.

By default, users of your application can use the OVERRIDE command to override the error condition that is caused by attempting to leave an observation without providing a value for a required field. You can prevent this by indicating in the **Override on required** field of the FSEDIT Parms window that overriding is not allowed. See "Modifying General Parameters" on page 57 for details.

CAUTION:
Do not assign this attribute to a field that is also assigned the PROTECTED attribute. Doing so would require users of your application to enter a value in a field that does not permit data entry. △

CAPS
specifies fields in which entered values are to be automatically capitalized (converted to all uppercase characters). This attribute has no effect on fields for numeric variables.

Type a **C** in the first position of a field to specify that the field value is to be automatically capitalized. To enable lowercase letters to remain lowercase in a field for which the CAPS attribute is currently specified, type an underscore or a blank space over the **C** in the field.

The CAPS attribute is set for all fields if you create a new data set using the NEW= option on PROC FSEDIT. If the data set already exists and you are just creating a new custom screen, FSEDIT looks in the first observation to determine whether to set the CAPS attribute.

If a custom screen is not specified in the procedure, or has been specified but has not been saved yet, the default setting for the CAPS attribute for each variable is determined by the value of the corresponding variable in the first observation of the data set. If a variable's value in the first observation contains lowercase characters, then capitalization is turned off for that variable's field; otherwise, capitalization is left on. Although the CAPS attribute is on by default for all numeric variables, this attribute has no effect on numeric variables.

FCOLOR
specifies the text color for each field. If the user's device does not support extended color attributes, this information is ignored.

Type the character that corresponds to the desired color in each field of this frame. (See "Codes for Color and Highlighting Attributes" on page 56.) The initial color code for all fields is **Y** (yellow).

ECOLOR
specifies the text color that will be used for each field when an error condition involving the field is detected. You can use this attribute to draw attention to data entry errors. If the user's device does not support extended color attributes, this information is ignored.

Type the character that corresponds to the desired color in each field of this frame. (See "Codes for Color and Highlighting Attributes" on page 56.) The initial color code for all fields is **R** (red).

FATTR

specifies the text highlighting attribute of each field.

Type the character that corresponds to the desired highlighting attribute in each field of this frame. (See "Codes for Color and Highlighting Attributes" on page 56.) There is no default highlighting attribute.

EATTR

specifies the text highlighting attribute that will be used for each field when an error condition involving the field is detected. You can use highlighting to draw attention to data entry errors. If the user's device does not support extended highlighting attributes, this information is ignored.

Type the character that corresponds to the desired highlighting attribute in each field of this frame. (See "Codes for Color and Highlighting Attributes" on page 56.) The initial highlighting attribute code for all fields is **H** (high intensity).

PAD

specifies which character is used to display fields in which no value has been entered.

Type the desired pad character in the first position of each field of this frame. (After you press ENTER, all positions in the field are filled with the specified pad character.) The initial pad character for all fields is the underscore (_).

When the FSEDIT procedure processes a value that is entered in a padded field, it converts any pad characters that remain in the field to blanks. Therefore, it is best to choose a pad character that is not likely to be contained in a value for that field.

Note: To include pad characters in field values, you can edit the field value after initial data entry. For example, if you enter an underscore character in a field that is padded with underscores, the entered underscore is converted to a blank when the value is processed. However, padding is not used after a value is entered in the field, so you can then immediately edit the field value to restore the desired underscore. △

PROTECT

specifies whether fields are protected. Values in protected fields in existing observations cannot be changed. When new observations are added, values cannot be entered in protected fields.

Type a **P** in the first position of a field to protect the field.

CAUTION:

Do not assign this attribute to a field that is also assigned the REQUIRED attribute. Doing so would require users of your application to enter a value in a field that does not permit data entry. △

JUSTIFY

specifies the alignment of values in fields.

Type one of the following values in each field:

L	aligns values against the left side of the field.
R	aligns values against the right side of the field.
C	centers values in the field.

If you leave a field in this frame blank, the corresponding field in the application display is right-aligned if it is a numeric field or left-aligned if it is a character field (unless the $CHAR. format is used).

NONDISPLAY

specifies fields in which values are not to be visible. This attribute does not prevent values from being entered in a field; it prevents values that are typed in a

field from appearing on the display. This attribute is useful for protecting fields that contain passwords or other sensitive information.

Type an **N** in the first position of a field to prevent values from being displayed in the corresponding field of the application display.

NOAUTOSKIP
specifies fields that the cursor does not leave unless it is explicitly moved. By default, when the user types a character in the last position of a field, the cursor jumps to the first position in the next field. When this attribute is specified, the cursor does not automatically jump to the next field.

Type an **N** in the first position of a field of this frame to prevent the cursor from automatically jumping from that field to the next field of the application display.

NOAUTOBLANK
specifies which numeric fields are not automatically blanked. This attribute is ignored for character fields.

Type an **N** in the first position of a field to prevent the automatic blanking of characters following the first blank in corresponding numeric fields in the FSEDIT window.

By default, when the FSEDIT procedure processes the values that users enter in numeric fields, it automatically clears all character positions following the first blank that is encountered in the fields. This is a useful feature in most fields because it enables users to enter numeric values left-justified in the field without having to manually blank out the remainder of the field. (Values in numeric fields are right-justified by default.) However, some numeric informats allow values that contain embedded blanks. Examples include date informats such as DATE*w.* and MMDDYY*w.*, as well as the BZ*w.d* informat. For fields that use these informats, you can specify the NOAUTOBLANK attribute to suppress the automatic blanking feature so that users can enter values that contain blanks.

Codes for Color and Highlighting Attributes

The following codes are valid for the FCOLOR and ECOLOR field attributes:

B	blue	G	green	W	white	A	gray
R	red	C	cyan	K	black	N	brown
P	pink	Y	yellow	M	magenta	O	orange

When your application is used, the color attributes are ignored if the user's device does not support color. If you specify a color that is not available on the user's device, the procedure substitutes the available color that most closely matches the specified color.

The following codes are valid for the FATTR and EATTR field attributes:

H	high intensity
U	underlining
R	reverse video
B	blinking

Most monochrome devices support only high intensity and underlining. If a user's device does not support the highlighting attributes that you specify, the highlighting attribute assignments are simply ignored. Therefore, you can assign these field

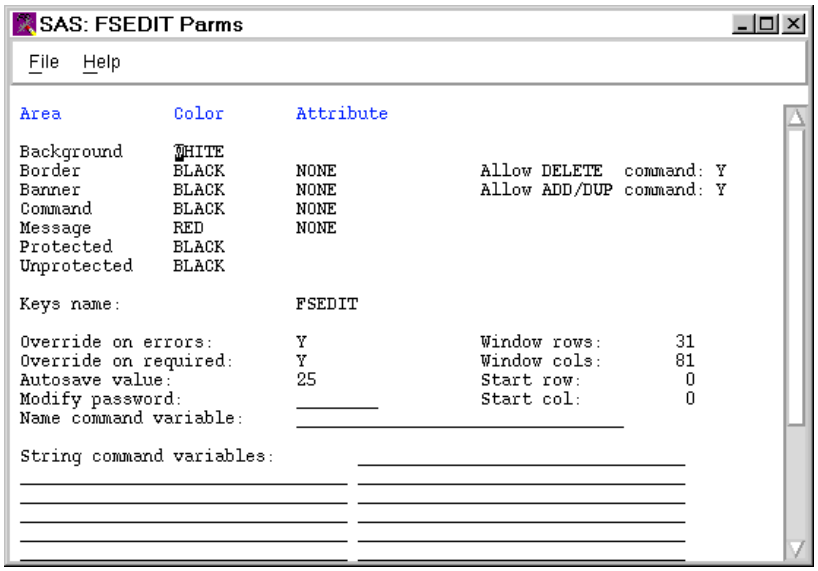
attributes even though the application may not always be used on a device that enables users to take advantage of color and highlighting.

Modifying General Parameters

Select option 5 from the FSEDIT Menu window to view or modify the current parameter settings for your FSEDIT application. This option opens the FSEDIT Parms window. Use the END command to close the FSEDIT Parms window and return to the FSEDIT Menu window.

Display 4.5 on page 57 shows the FSEDIT Parms window for a typical application.

Display 4.5 The FSEDIT Parms Window



Parameter Fields

To change one of the general parameters of the FSEDIT session, modify the value in the corresponding parameter field in the FSEDIT Parms window. (If the field has a current value, simply type over it.)

Here are descriptions of the available parameter fields:

Color and Attribute

On devices that support color, you can change the default color and highlighting attribute of the following window areas:

Background — controls the background color of the FSEDIT window.

Note: Some devices do not allow the background color to be changed; for these devices, the background color parameter is ignored. △

Border — controls the color of the window border in character-based display environments.

Note: This parameter has no effect in graphical windowing environments. △

Banner — controls the color of the **Command===>** text at the left of the command line.

Note: This parameter has no effect if a menu bar is displayed in place of a command line. △

Command — controls the color of the text that users type on the command line.

Note: This parameter has no effect if a menu bar is displayed in place of a command line. △

Message — controls the color of text that is displayed in the window's message line.

Protected — controls the color of all descriptive text in the FSEDIT window.

Unprotected — controls the color of all variable fields in the FSEDIT window—even those for which the PROTECT field attribute is specified in the FSEDIT Attribute window.

Note: If you change the color values in the **Protected** and **Unprotected** fields of this window, the specified colors override the colors that you used when you created the custom display for the application in the FSEDIT Modify window. △

The following values are valid in the **Color** fields:

BLUE	GREEN	WHITE	GRAY
RED	CYAN	BLACK	BROWN
PINK	YELLOW	MAGENTA	ORANGE

Note: If a specified color is not available on a user's display device, the procedure substitutes the available color that most closely matches the specified color. △

On devices that support extended highlighting attributes, you can assign a highlighting attribute to specified areas in the window. (The **Background**, **Protected**, and **Unprotected** areas do not support highlighting attributes.) The following values are valid in the **Attribute** fields:

NONE — no highlighting

HIGHLIGHT — high intensity

BLINKING — blinking

UNDERLINE — underlining

REVERSE — reverse video

Note: If a parameter specifies a highlighting attribute that is not available on the user's display device, the parameter is ignored. △

Allow DELETE command
controls whether users can issue the DELETE command to delete observations from the data set.

Specify **Y** in this field (or leave it blank) if you want to permit users to delete observations from the displayed data set. (You can use the NODEL option with the PROC FSEDIT statement to override this parameter setting when the FSEDIT

session is invoked.) Specify **N** in this field to disable the DELETE command in the FSEDIT window.

Allow ADD/DUP command
controls whether users can issue the ADD or DUP commands to add new observations to the data set.

Specify **Y** in this field (or leave it blank) if you want to permit users to add new observations to the displayed data set. (You can use the NOADD option with the PROC FSEDIT statement to override this parameter setting when the FSEDIT session is invoked.) Specify **N** in this field to disable the ADD and DUP commands in the FSEDIT window.

Keys name
identifies the KEYS entry that contains function key assignments for the application. The default is FSEDIT, which selects the default entry FSEDIT.KEYS. The FSEDIT procedure searches for the specified entry in the following catalogs in the order shown:

1. the SAS catalog that is identified in the SCREEN= option of the PROC FSEDIT statement or in the *screen-name* parameter of the FSEDIT command
2. SASUSER.PROFILE (or WORK.PROFILE if the SASUSER library is not allocated)
3. SASHELP.FSP

If the specified entry is not found, the default FSEDIT key definitions are used.

Override on errors
determines whether users of your application are permitted to use the OVERRIDE command to exit an observation even though one or more fields contain invalid values (such as a value that is outside the acceptable range that is assigned by the MINIMUM and MAXIMUM attributes).

Specify **Y** to permit the use of the OVERRIDE command in these situations. Specify **N** to prevent users from exiting an observation without supplying a value within an acceptable range. **Y** is the default.

Override on required
determines whether users of your application are permitted to use the OVERRIDE command to exit an observation even though one or more fields that have been assigned the REQUIRED attribute contain no value.

Specify **Y** to permit the use of the OVERRIDE command in these situations. Specify **N** to prevent users from exiting an observation without supplying values for all required fields. **Y** is the default.

Autosave value
determines how frequently the data set that the application uses is automatically saved. By default, AUTOSAVE is set to 25, which means that the data set is automatically saved after each group of 25 observations is entered, edited, or deleted.

You can also set the AUTOSAVE parameter by using the AUTOSAVE command.

Modify password
enables you to protect your customized application by assigning a password to it. If you assign a value in the **Modify password** field, then users of the application must specify the password with the MODIFY command in order to modify the SCREEN entry. Also, if you assign a password, the MODIFY option is no longer valid in the PROC FSEDIT statement.

The password can consist of any combination of letters and numbers, but it must begin with a letter.

Rows and columns
: enables you to specify the height and width (in rows and columns) of the FSEDIT window for your application. You can position the FSEDIT window within the display by specifying the row and column for the upper left corner of the window.

Name command variable
: enables you to assign a default variable to search with the LOCATE command in your application. If you specify a search variable here, users do not have to issue a NAME command before using the LOCATE or LOCATE: commands in the FSEDIT window. The search variable must be a data set variable. Computed variables cannot be used as search variables.

String command variables
: enables you to specify up to 29 variables to search for embedded text with the SEARCH command in your application. If you specify search variables here, users do not have to issue a STRING command before using the SEARCH or SEARCH@ commands in the FSEDIT window. The search variables must be data set variables. Computed variables cannot be used as search variables.

Commands Versus Parameter Settings

Values for the NAME, STRING, and AUTOSAVE parameters, which are described above, are saved when the FSEDIT Menu window is closed. Users can override the stored parameter values for the duration of an FSEDIT session by executing the NAME, STRING, or AUTOSAVE commands in the FSEDIT window. If a user opens the FSEDIT Menu during an FSEDIT session, then any changes that are made with the NAME, STRING, and AUTOSAVE commands in the FSEDIT window are automatically saved with the customized information.

Creating Application-Specific Key Definitions

The FSEDIT procedure enables you to specify a customized set of function key assignments. This gives you control over which commands the function keys issue in your application.

By default, the FSEDIT procedure uses the function key assignments that are defined in the FSEDIT.KEYS entry in the SASHELP.FSP catalog. This is one of the standard catalogs that are defined automatically when a SAS session is initiated. The SASHELP.FSP catalog is shared by all SAS users at your site, so when you use the KEYS command in the FSEDIT window, the procedure creates a copy of the FSEDIT.KEYS entry in your personal PROFILE catalog (SASUSER.PROFILE, or WORK.PROFILE if the SASUSER library is not allocated). This copy is then used in subsequent FSEDIT sessions.

You can use the KEYS= option with the PROC FSEDIT statement to select a different KEYS entry for your FSEDIT session. When you use the KEYS= option, the procedure searches the following catalogs in the order shown for the specified KEYS entry:

1. the SAS catalog that is identified in the SCREEN= option, if that option was also used with the PROC FSEDIT statement
2. SASUSER.PROFILE (or WORK.PROFILE if the SASUSER library is not allocated)
3. SASHELP.FSP

If the specified KEYS entry is not found, a blank KEYS entry that has the specified name is created in the catalog that is identified in the SCREEN= option, or in your

personal PROFILE catalog if the SCREEN= option was not used with the PROC FSEDIT statement.

When you use the MODIFY command to create a new SCREEN entry, the KEYS entry that is used when the SCREEN entry is created is recorded in the **Keys name** field in the FSEDIT Parms window. (See "Modifying General Parameters" on page 57 for details about the FSEDIT Parms window.) If you do not use the KEYS= option in the PROC FSEDIT statement that initiates the FSEDIT session, the KEYS entry name is FSEDIT.

Once a SCREEN entry is created, the KEYS entry name that is specified in the SCREEN entry parameter takes precedence over one that is specified in a KEYS= option. For example, assume that you have previously created a SCREEN entry named DISPLAY.SCREEN in the MASTER.CUSTOM catalog, and that the **Keys name** parameter that is specified in the SCREEN entry is MYKEYS. If you then submit the following statements, the procedure uses the KEYS entry MYKEYS.KEYS (specified in the SCREEN entry) rather than the EDKEYS.KEYS entry (specified in the KEYS= option).

```
proc fsedit data=master.subscrib
     screen=master.custom.display.screen
     keys=edkeys;
run;
```

The procedure looks for the KEYS entry MYKEYS.KEYS in the MASTER.CUSTOM catalog, then in SASUSER.PROFILE (or WORK.PROFILE), then in SASHELP.FSP. If the entry is not found, a blank KEYS entry is created.

Note: The FSEDIT NEW window always uses the KEYS entry that was specified in the KEYS= option or in the default entry FSEDIT.KEYS, not the entry that was specified in the SCREEN entry. △

You can change the associated KEYS entry during an FSEDIT session by using option 5 in the FSEDIT Menu window. Use the MODIFY command to open the FSEDIT Menu window; then select option 5 to open the FSEDIT Parms window. Enter the name of the desired KEYS entry in the **Keys name** field. The new value takes effect immediately. If the specified entry is not found in the current screen catalog (or in your PROFILE catalog or SASHELP.FSP), then a new blank KEYS entry is created in the screen catalog if one has been identified; otherwise, the entry is created in your personal PROFILE catalog.

CHAPTER

5 The FSLETTER Procedure

Overview

The FSLETTER procedure enables you to create, edit, and print letters and other documents. When creating and editing documents in the FSLETTER window, you can use all the features of the SAS text editor, including the spelling checker.

Your FSLETTER documents can include named fields. When the document is printed, you can fill in these fields manually, or the procedure can fill the fields automatically using values from a SAS data set. This is convenient for creating and maintaining form letters, for example. You can print individual copies of the document, or you can automatically print a copy for every observation in a data set.

You can enter the FSLETTER procedure from the FSBROWSE and FSEDIT procedures. See "FSEDIT Window Commands" on page 26 for details about the LETTER, EDIT, and SEND commands that enable you to do this.

Note: You can also open the FSLETTER window by issuing an FSLETTER command from any SAS System command line. △

FSLETTER Procedure Syntax

PROC FSLETTER LETTER=*SAS-catalog*<*.catalog-entry*>
<DATA=*data-set*>
<NOBORDER>
<PRINTFILE=*fileref* | *'actual-filename'*>;

WHERE *expression*;

The PROC FSLETTER statement is required. The WHERE statement is optional.

PROC FSLETTER also includes global commands that can be used after the procedure has been invoked (seeChapter 9, "SAS/FSP Software Global Commands," on page 143).

PROC FSLETTER Statement

Initiates the FSLETTER procedure and specifies the SAS catalog in which documents that are created with the procedure are stored.

Requirement: You must specify a SAS catalog when you initiate the FSLETTER procedure. Documents that are created using the procedure are stored in this catalog

Tip: Use the DATA= option to specify a data set that will be used to fill variable fields when the document is printed.

PROC FSLETTER LETTER=*SAS-catalog*<*.catalog-entry*>
<DATA=*data-set*>
<NOBORDER>
<PRINTFILE=*fileref* | '*actual-filename*'>;

Required Argument

You must use the LETTER= argument in the PROC FSLETTER statement. This argument identifies the catalog in which documents, forms, and editor parameters that are created using the FSLETTER procedure are stored. If the specified catalog does not exist, it is created. The argument can also be used to specify a particular entry to edit. The procedure terminates with an error message if this argument is omitted.

Valid forms of the LETTER= argument are

LETTER=*SAS-catalog*<*.catalog-entry*>

CATALOG=*SAS-catalog*<*.catalog-entry*>

CAT=*SAS-catalog*<*.catalog-entry*>

C=*SAS-catalog*<*.catalog-entry*>

The general form of the *SAS-catalog* value is

<*libref*.>*catalog-name*

If you specify a one-level name, it is treated as a catalog name in the default library, WORK. Remember that the contents of the WORK library are erased at the end of each SAS session. Use a two-level catalog name to permanently store your work. You must use a two-level catalog name if you want to specify a catalog entry name.

The general form of the *catalog-entry* value is

entry-name<*.entry-type*>

If you omit the catalog entry name, an Explorer window is opened showing the current contents of the specified catalog.

If you supply an entry name in addition to a catalog name, the procedure opens the appropriate window for editing the entry:

- If the entry type is LETTER, or if the entry type is omitted, an FSLETTER window is opened.
- If the entry type is FORM, a FORM window is opened.
- If the entry is FSLETTER.EDPARMS, an EDPARMS window is opened.

The procedure terminates with an error message if you specify an entry type that is not supported by the FSLETTER procedure.

Options

The following options can be used in the PROC FSLETTER statement:

DATA=*data-set*<(*data-set-options*)>
names an existing SAS data set that will be used to fill fields in the document. The FSLETTER procedure terminates with an error message if the specified data set does not exist.

Note: This option is valid only if you also specify the name of an existing LETTER entry in the LETTER= argument. △

When you use this option, the procedure does not initiate an interactive FSLETTER session. A pause occurs while a copy of the document is printed for each observation in the data set; then the procedure ends.

You can follow the data set name with a list of data set options enclosed in parentheses. The following data set options are valid with the DATA= option:

DROP	RENAME
KEEP	WHERE

See *SAS Language Reference: Dictionary* for descriptions of these data set options.

NOBORDER
suppresses the sides and bottom of the FSLETTER window's border in a character-based display environment.

Note: This option is ignored in graphical windowing environments. △

When this option is used in a supported display environment, text can appear in the columns and row that the border normally occupies.

PRINTFILE=*fileref* | '*actual-filename*'
PRTFILE=*fileref* | '*actual-filename*'
PRINT=*fileref* | '*actual-filename*'
DDNAME=*fileref* | '*actual-filename*'
DD=*fileref* | '*actual-filename*'
specifies the fileref or the actual name of an external file to which procedure output is directed. By default, procedure output is sent to the destination that is specified in the associated FORM entry. When this option is used, output is sent to the specified file instead.

If you use a fileref, you must have previously assigned the fileref to an external file with the FILENAME statement. If you use an actual filename, enclose it in quotes.

WHERE Statement

Specifies a condition or set of conditions that observations in an input data set must meet in order to be used for filling variable fields.

Restriction: The WHERE statement is valid only when the DATA= option is used in the PROC FSLETTER statement to identify the data set.

WHERE *expression*;

Argument

expression is any valid WHERE expression that includes one or more of the variables in the input data set. Refer to the description of the WHERE statement in *SAS Language Reference: Dictionary* for details about the operators and operands that are valid in WHERE expressions.

Using the WHERE Statement

By default, when you use the DATA= option in the PROC FSLETTER statement, the FSLETTER procedure generates a copy of the document for every observation in the data set. If you do not need a copy for every observation, you can use the WHERE statement to generate copies for only observations that meet a specified condition or list of conditions.

For example, to print form letters for only observations in which the value of the variable YEAR is less than 5, follow the PROC FSLETTER statement with this statement:

```
where year<5;
```

In this case, the FSLETTER procedure uses only observations that meet the specified condition. No documents are printed for observations that do not satisfy the condition.

FSLETTER Command Syntax

Tip: The FSLETTER command provides an easy way to initiate an FSLETTER session from any SAS System command line.

FSLETTER <? | *SAS-catalog*<*.catalog-entry*>>

FSLETTER Command

Initiates an FSLETTER session.

FSLETTER <? | *SAS-catalog*<*.catalog-entry*>>

Arguments

The following arguments can be used with the FSLETTER command:

?
opens a selection window from which you can choose the catalog that is used by the FSLETTER procedure. You can select the catalog interactively by pointing and clicking in the selection lists, or you can specify the desired catalog explicitly in the **Member Name** field. Once you specify the catalog, an Explorer window opens showing the contents of the specified catalog. From the Explorer window, you can select an existing entry or create a new entry.

SAS-catalog<.catalog-entry>
specifies the catalog in which documents, forms, and editor parameters for the FSLETTER session are stored and, optionally, the specific entry. The complete form of the argument is

<libref.>catalog-name<.entry-name<.entry-type>>

You can specify a one-, two-, three-, or four-level name:

- If a one-level name is specified, it is treated as a catalog name in the default library, WORK. If the specified catalog does not already exist in the WORK library, it is created.

  Remember that all catalogs in the WORK library are erased when you end your SAS session.
- If a two-level name is specified, it is treated as *libref.catalog-name*. If the specified catalog does not already exist in the specified library, it is created.
- If a three-level name is specified, it is treated as *libref.catalog-name.entry-name*. The entry type is assumed to be LETTER. If the specified catalog does not already exist in the specified library, it is created.
- If a four-level name is specified, the fourth level should be EDPARMS, FORM, or LETTER. Invalid catalog entry types are ignored; LETTER is used instead. If the specified catalog does not already exist in the specified library, it is created.

If you use a one- or two-level name, the procedure initially opens an Explorer window, from which you can select an existing entry or create a new entry in the specified catalog.

If you use a three- or four-level name, the procedure opens the appropriate window for creating or editing the specified entry:

- If the entry type is LETTER, or if the entry type is omitted, an FSLETTER window is opened.
- If the entry type is FORM, a FORM window is opened.
- If the entry is FSLETTER.EDPARMS, an EDPARMS window is opened.

The procedure terminates with an error message if you specify an entry type that is not supported by the FSLETTER procedure.

If you do not specify any of these arguments, the selection window that was described for the **?** argument is opened.

Using the FSLETTER Command

The FSLETTER command is a convenient way to open the FSLETTER window because it can be issued in any SAS System window. When you end an FSLETTER session that was initiated with the FSLETTER command, you return to whatever window was active when the command was issued.

CHAPTER

6

FSLETTER Procedure Windows

Overview

The process of producing a document with the FSLETTER procedure involves the following steps:

1 *Selecting catalog entries.*

The FSLETTER procedure provides two ways to choose which catalog entry to create or edit:

- You can specify a complete catalog entry name in the LETTER= option of the PROC FSLETTER statement or in the FSLETTER command. In this case, the FSLETTER window is opened for creating or editing the specified entry.
- You can specify only a catalog name in the LETTER= option of the PROC FSLETTER statement or in the FSLETTER command. In this case, an Explorer window is opened. From this window you can view the current entries in the specified catalog, select entries to browse or edit, or create new entries. You can also print existing documents from the Explorer window.

2 *Creating and editing documents.*

Documents are created and edited in the FSLETTER window. This window uses the SAS text editor, which provides a variety of commands and features that simplify the process of entering and editing the text of your document.

You can define fields in your document that enable you to enter information that changes from one copy of the document to the next. Fields are filled when the document is printed.

3 *Assigning field attributes.*

If you include fields in your document, you can open the FSLETTER ATTR window from the FSLETTER window to define the attributes of any fields in the document. Field attributes control whether field values are given special handling when the document is printed.

4 *Printing documents.*

Documents are prepared for printing in the FSLETTER SEND window. In this window you fill any fields in the document and make final editing changes before sending the document to a printer or to an external file.

Note: The process is different if you use the DATA= option in the PROC FSLETTER statement. In that case, the procedure is not interactive; it prints a copy of the document for each observation in the specified data set without opening any windows. Unless you use a WHERE statement in conjunction with the PROC FSLETTER statement, a copy is produced for every observation in the data set. The FSLETTER procedure ignores the FIRSTOBS= and OBS= system options. △

You can also create FORM entries and EDPARMS (editor parameter) entries with the FSLETTER procedure. The process of creating these entries involves the first two of the steps listed above, except that these entry types use their own special windows rather than the FSLETTER window:

- FORM entries are created and modified in the FORM window.
- EDPARMS entries are created and modified in the EDPARMS window.

The following sections describe the various FSLETTER procedure windows and the commands that are valid in each.

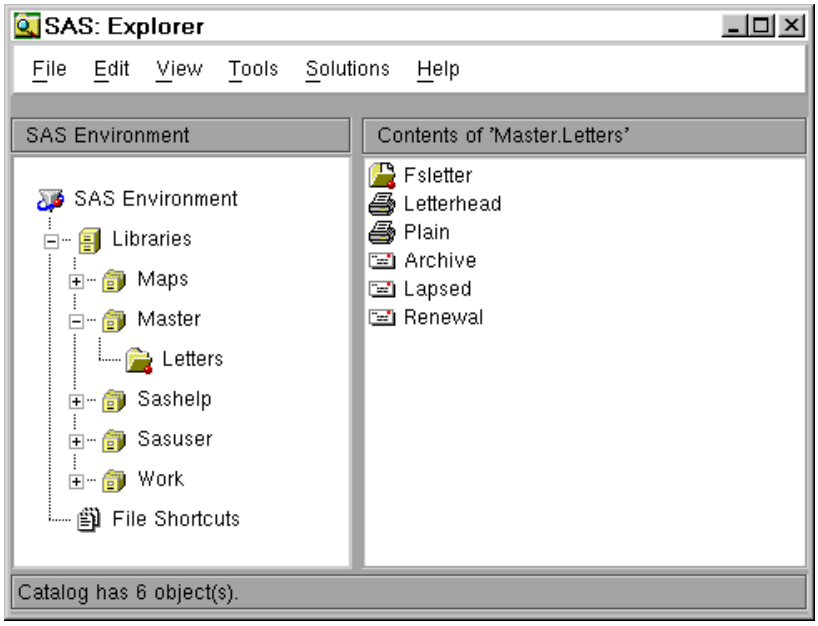
Selecting Catalog Entries

You can select the entry to create or edit by specifying a catalog entry name with the catalog name in the LETTER= option of the PROC FSLETTER statement or in the FSLETTER command. In that case, the initial window in the FSLETTER session is the appropriate window for the type of entry (FSLETTER, FORM, or EDPARMS).

If you specify only a catalog name when you invoke the FSLETTER procedure, then the initial window in the FSLETTER session is an Explorer window. The Explorer window enables you to manipulate catalog entries in various ways. You can create new entries or display existing entries for editing, browsing, or printing. You can also rename, delete, and copy entries.

Display 6.1 on page 71 shows an Explorer window for a catalog that contains LETTER, FORM, and EDPARMS entries.

Display 6.1 Typical FSLETTER Catalog Displayed in the Explorer Window



You can select an existing entry for editing by double-clicking on the corresponding item in the Explorer window, or by using the TAB or cursor keys to select the desired entry and then pressing ENTER. To create a new entry, issue the NEWOBJ command or select

File ▶ New

Select the desired entry type from the resulting list, then specify a name for the new entry.

For more information about using the Explorer window, refer to the online Help for Base SAS software.

Creating and Editing Documents

Documents are entered or edited in the FSLETTER window. FSLETTER windows can be opened for editing in the following ways:

- by specifying the name of a LETTER entry in the LETTER= argument of the PROC FSLETTER statement or in the FSLETTER command
- by selecting an existing LETTER entry or creating a new LETTER entry in the Explorer window
- by issuing an EDIT command in another FSLETTER window
- by issuing an EDIT command in an FSBROWSE or FSEDIT window.

If the LETTER entry that you specify when you open the FSLETTER window is not in the current catalog, an empty FSLETTER window is opened for entering text. If the specified entry already exists, it is displayed for editing. All text in the FSLETTER window is unprotected. Using editing keys and commands, you can write new documents and edit existing documents.

The FSLETTER window uses the SAS text editor, so all the standard text editor commands are available. See the online Help for Base SAS software if you need an

introduction to the features of the SAS text editor. You can customize many features of the text editor to suit your own tastes and needs. Refer to "Setting Text Editor Parameters" on page 86 for a discussion of the features that you can control.

Choosing a Form

When you create a new document, you should choose a form to associate with it. *Forms* are SAS catalog entries of type FORM that contain instructions for formatting the text in your document when it is printed. In addition, the line length specified in the form determines the width of the text entry area in the FSLETTER window.

Newly created documents are assigned the default form FSLETTER.FORM. You can change the form assignment by using the FORM command in the FSLETTER window. When you change the form assignment, the procedure first looks for the new form in the current catalog (the catalog in which the current document resides). If the specified entry is not there, the procedure looks in your personal PROFILE catalog (SASUSER.PROFILE, or WORK.PROFILE if the SASUSER library is not allocated). If the specified entry is not there, the procedure looks last in the SASHELP.FSP system catalog. If the specified entry still is not found, an error message is returned, and the current form assignment is not changed.

Defining Fields

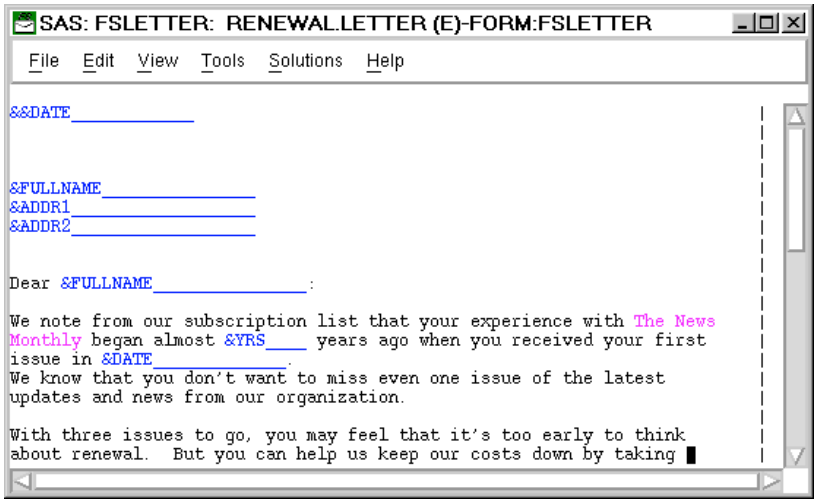
The FSLETTER procedure enables you to define *fields* in your documents. Fields are areas in which values can change for each copy of a document. A field appears in the text of a document as an ampersand (&) followed by a valid SAS name.

The length of each field in your document should be equal to the largest number of characters expected for that value. The field length is determined by the number of characters in the field name (including the ampersand). To extend a field, you can follow the field name with a series of underscores (_). The underscores increase the field length, but they are not part of the field name.

You can use the automatic variables &DATE and &DATE7 as fields in FSLETTER documents. These variables provide the current date when the document is printed. Because the variable names already begin with an ampersand, the field names should begin with two ampersands (**&&DATE** and **&&DATE7**). The variables provide the current date in WORDDATE. and DATE7. formats, respectively. If you use **&&DATE**, add underscores to extend the field to 18 characters (the maximum possible length for date values in the WORDDATE. format).

Display 6.2 on page 73 shows a portion of a document displayed in the FSLETTER window. This portion of the document contains six different fields; **FULLNAME** appears twice.

Display 6.2 FSLETTER Document Containing Fields



You define fields in the FSLETTER window, but you do not supply values for the fields until you print the document from the FSLETTER SEND window. See "Printing Documents" on page 84 for details. The FSLETTER procedure can also fill the fields automatically, using values from observations in a SAS data set. See "Step 1: Filling Fields" on page 85 for details.

Using Color and Highlighting Attributes

If your terminal or workstation supports color and highlighting, then you can change the color and highlighting of characters as you type (or overtype) them. Use the global COLOR TEXT command to change the color and highlighting attributes of the text that you enter. For example, the following command changes all the text you type after the command is issued to reverse-video magenta:

```
color text magenta reverse
```

Once you enter a COLOR TEXT command, the specified attributes are used until you change them with another COLOR command. Refer to the description of the COLOR command in the online Help for Base SAS software for additional details.

Note: Some terminals or workstations provide special keys that control text color and highlighting. If your device has such keys, you can use them to set color and highlighting attributes as you enter the text. △

The color and highlighting attributes are stored with the document; therefore, they remain in effect the next time the document is displayed.

You can use color and highlighting in FSLETTER documents

- to enhance displayed text.

  You can change the default text color and highlighting attributes to change the appearance of the document in the FSLETTER and FSLETTER SEND windows. For example, you can use a different color or highlighting attribute for the fields in the document so that they stand out more clearly when the text is displayed in the FSLETTER SEND window for data entry.

- to signal printing instructions.

  The form that is associated with the document may interpret certain colors or highlighting (or combinations of color and highlighting) as signals to use a

particular printer feature when the document is printed. For example, the form can interpret the combination of red and reverse video as a signal that text with those attributes should be underlined or italic when printed.

You can edit the FORM entry to select which attributes signal printing instructions for your documents. For details about how the color and highlighting attributes that signal printing instructions are defined in the FORM entry, see the discussion of the FORM window in the online Help for Base SAS software.

You can use the global FONT command in the FSLETTER window to check which color and highlighting attributes are defined as printing instructions for the current form.

FSLETTER Window Commands

In addition to the SAS text editor commands that are described in the online Help for Base SAS software and the global commands described in Chapter 9, "SAS/FSP Software Global Commands," on page 143, you can use the following commands when editing a document in the FSLETTER window:

Managing LETTER entries

ATTR

COPY <*libref.catalog-name*>*entry-name*<.LETTER>

DES *description*

EDPARMS

FORM *form-name*

SAVE *letter-name*

SEND <ALL> <DATA=*data-set*<(*data-set-options*)>>
<FILE=*fileref* | '*actual-filename*'<APPEND | REPLACE>>

Opening Additional Windows

BROWSE *letter-name*<.LETTER> | *form-name*.FORM | FSLETTER.EDPARMS

EDIT *letter-name*<.LETTER> | *form-name*.FORM | FSLETTER.EDPARMS

Closing the Window

CANCEL

END

Managing Data Sets

CLOSE

DATA <*data-set*<(*data-set-options*)>>

WHERE <<ALSO> *expression*> | <UNDO | CLEAR>

Command Descriptions

Descriptions of the FSLETTER window commands:

ATTR
opens the FSLETTER ATTR window, in which the field attributes for the current document can be viewed or modified.

Assigning an attribute to a field tells the FSLETTER procedure to take certain actions when it encounters the field in the FSLETTER SEND window. See "Assigning Field Attributes" on page 79 for more information.

BROWSE *letter-name*<.LETTER> | *form-name*.FORM | FSLETTER.EDPARMS
opens an additional window for browsing the specified entry:

- If the entry type is LETTER, or if no entry type is specified, another FSLETTER window is opened for browsing the specified document.
- If the entry type is FORM, a FORM window is opened for browsing the specified form. For information on using the FORM window, see the online Help for Base SAS software.
- If the entry is FSLETTER.EDPARMS, an EDPARMS window is opened for browsing the default editor parameter file. For information on using the EDPARMS window, see "Setting Text Editor Parameters" on page 86.

An error occurs if you specify an entry type that is not supported by the FSLETTER procedure.

The new window that is opened by the BROWSE command becomes the active window. You can use the SWAP command to move between the windows that have been opened in the current FSLETTER session. Use the END command to close the new window and return to the previous window.

CANCEL
cancels all changes that have been made to the current document since the FSLETTER window was opened (or since the last SAVE command), closes the current FSLETTER window, and returns you to the window from which the FSLETTER window was opened.

CLOSE
closes the data set that was opened with the DATA command. After the data set is closed, the FSLETTER procedure no longer uses it to fill fields during printing.

COPY <*libref.catalog-name*>*entry-name*<.LETTER>
copies the specified LETTER entry into the current document. You can use text editor line-target commands (A or B, for after or before) to copy the document at a specific position within the current text. Otherwise, the copied document is appended to the end of the current document.

If the document being copied uses a different form than the current document, the document being copied is flowed (if necessary), using the line length information in the current document's form.

DATA <*data-set*<(*data-set-options*)>>
opens the specified SAS data set for filling fields in documents.

While the data set is open, any fields in a document whose names or aliases match variable names in the specified data set are filled automatically from observations in the data set when the document is printed.

You can add a list of data set options following the data set name. The list must be enclosed in parentheses. The following data set options are valid with the DATA command:

DROP	RENAME
KEEP	WHERE

Refer to *SAS Language Reference: Dictionary* for descriptions of these data set options.

Use the CLOSE command to close the data set that was opened with the DATA command if you no longer want fields to be filled from the data set.

DES *description*
assigns a description of up to 40 characters to a document. The description does not need to be enclosed in quotes. The description is displayed with the name of the document when you select the Details view in the Explorer window.

EDIT *letter-name*<.LETTER> | *form-name*.FORM | FSLETTER.EDPARMS
opens an additional window for creating or editing the specified entry:

- If the entry type is LETTER, or if no entry type is specified, an FSLETTER window is opened for editing the specified document. If the specified entry does not exist, the FSLETTER window is initially blank. The entry is not created until you enter text and save the document or close the FSLETTER window. (No entry is created when you close a blank FSLETTER window.)
- If the entry type is FORM, a FORM window is opened for creating or editing the specified form. For information on using the FORM window, see the online Help for Base SAS software.
- If the entry is FSLETTER.EDPARMS, an EDPARMS window is opened for creating or editing the default editor parameter file. For information on using the EDPARMS window, see "Setting Text Editor Parameters" on page 86.

  Note: The parameter settings in the FSLETTER.EDPARMS entry do not affect the current document. The FSLETTER.EDPARMS entry determines the initial parameters for newly created documents. Use the EDPARMS command to change parameter settings for the current document. △

An error occurs if you specify an entry type that the FSLETTER procedure does not support.

The new window that is opened by the EDIT command becomes the active window. You can use the SWAP command to move between the windows that have been opened in the current FSLETTER session. Use the END command to close the new window and return to the previous window.

EDPARMS
opens an EDPARMS window and displays the text editor parameter settings for the current document. Any changes that you make take effect when the EDPARMS window is closed. The parameter settings are stored in the LETTER entry and remain in effect the next time you edit this document.

See "Setting Text Editor Parameters" on page 86 for more information on using the EDPARMS window.

Note: Changes made in the EDPARMS window that is opened by the EDPARMS command do not affect the default parameter settings in the FSLETTER.EDPARMS entry for the catalog. △

END
saves the currently displayed document, closes the FSLETTER window, and returns you to the window from which the FSLETTER window was opened.

FORM *form-name*
changes the FORM entry that is assigned to a document. The form name is recorded in the LETTER entry.

When you specify a new form name, the FSLETTER procedure looks first for a FORM entry that has the specified name in the current catalog (the catalog that was opened when the procedure was initiated). If the form is not there, the procedure looks for it in your personal PROFILE catalog (SASUSER.PROFILE, or WORK.PROFILE if the SASUSER library is not allocated). If the form is not there, the procedure looks for it in the SASHELP.FSP system catalog. If the form still is not found, the procedure returns an error message, and the current form assignment is not changed.

If the new form specifies a different line width than the current form uses, a requestor window appears asking whether you want the document to be flowed to the new line width. Respond with a Y to flow the text or with an N to wrap the text. If you choose N, lines of text are split if the new line length is smaller, or blanks are added to pad the lines if the new line length is larger.

SAVE <*letter-name*>
stores the current version of the document in the catalog, leaving the document displayed for further editing. If you do not specify a name for the entry, the text is saved under its original name in the catalog. Specify a different name to store the current document under another name in the catalog.

It is wise to issue SAVE commands occasionally while entering text to avoid losing work due to interruptions in computing services. Any document that has been modified is automatically saved when you use the END or SEND commands.

SEND <ALL> <DATA=*data-set*<(*data-set-options*)>>
<FILE=*fileref* | '*actual-filename*' <APPEND | REPLACE>>
closes the FSLETTER window and prints the current LETTER entry. By default, the SEND command opens the FSLETTER SEND window, in which you can fill in fields and make final editing changes before the document is printed.

If a SAS data set is currently open, any fields in the document whose names or aliases match variable names in the open data set are filled automatically with values from the data set. If the data set was opened with a DATA command, the field values come from the first observation in the data set. If the data set is open because the FSLETTER window was opened by an EDIT command in an FSBROWSE or FSEDIT session, then the field values come from the observation that was displayed in the FSBROWSE or FSEDIT window when the EDIT command was issued.

If you include the ALL argument with the SEND command, a copy of the document is produced for every observation in the open data set (or for every observation that satisfies the WHERE condition, if a WHERE command has been issued). When you use the ALL option, the FSLETTER SEND window is not opened.

Note: The ALL argument is valid only when a data set is currently open. △

You can also use the DATA= option in the SEND command to produce a copy of the document for each observation in a specified data set. The DATA= option is useful in the following circumstances:

- when you want to produce copies of an individual document without using the DATA command to open a data set for the FSLETTER session.

- when you want to produce copies of an individual document for a data set other than the data set that is currently open. (The DATA= option has no effect on any other data set that may currently be open.)

Because the DATA= option produces a copy of the document for each observation in the data set, it is not necessary to use the ALL option with the DATA= option. When you use the DATA= option, the FSLETTER SEND window is not opened.

You can add a list of data set options following the data set name in the DATA= argument. The list must be enclosed in parentheses. The following data set options are valid with the DATA= option:

DROP	RENAME
KEEP	WHERE

Refer to *SAS Language Reference: Dictionary* for descriptions of these data set options.

Unless you issue a PRTFILE command or use the PRINTFILE= option in the PROC FSLETTER (or PROC FSBROWSE or PROC FSEDIT) statement that initiates the session, output is sent to the destination that is specified in the form associated with the document. To route output to an external file instead, use the FILE= option in the SEND command. The FILE= option is useful in the following situations:

- when you want to send an individual document to an external file while leaving the printer as the default output destination
- when you want to send an individual document to an external file other than the currently designated print file.

To identify the external file, you can use either a fileref or its actual filename. If you use a fileref, you must have previously assigned the fileref to the external file with a FILENAME statement. If you use an actual filename, enclose it in quotes.

By default, each new document that is sent to the file replaces any text that was previously in the file. (If you use the ALL or DATA= options to produce copies of a document for observations in a SAS data set, all copies are sent to the same file.) To append the output from the current SEND command to the existing text instead of overwriting the text, add the APPEND option to the SEND command. The REPLACE option specifies the default behavior.

WHERE <<ALSO> *expression*> | <UNDO | CLEAR>
imposes one or more sets of conditions that observations in the data set must meet in order to fill fields in printed documents. *Expression* is any valid WHERE expression that includes one or more of the variables in the input data set. (Refer to the description of the WHERE statement in *SAS Language Reference: Dictionary* for details about the operators and operands that are valid in WHERE expressions.) No documents are created for observations that do not satisfy the specified conditions.

Note: The WHERE command is valid only when a data set is open for the FSLETTER session. △

The complete set of conditions imposed by the WHERE command is called a temporary WHERE clause. The conditions can be modified or canceled during the FSLETTER session.

The WHERE command has the following forms:

WHERE *expression*

applies the conditions specified in *expression* as the new temporary WHERE clause, replacing any clause that was previously in effect.

WHERE ALSO *expression*
adds the conditions specified in *expression* to the existing temporary WHERE clause.

WHERE UNDO
removes the most recently added set of conditions from the temporary WHERE clause.

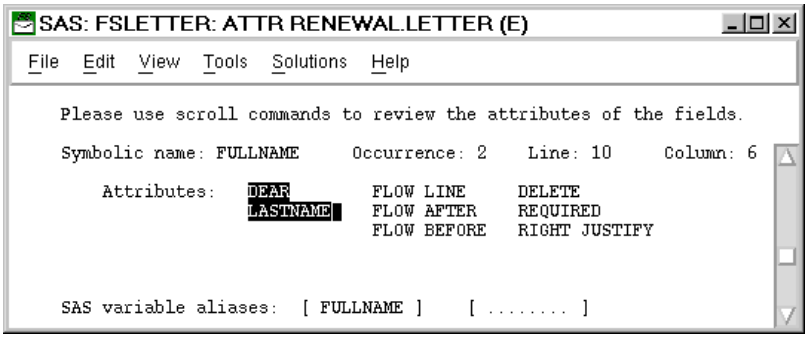
WHERE
WHERE CLEAR
cancels the current temporary WHERE clause.

Assigning Field Attributes

After entering the document text and defining variable fields, you can use the FSLETTER ATTR window to assign attributes to each field. Use the ATTR command in the FSLETTER window to open the FSLETTER ATTR window. Although you assign field attributes while editing the document, the attributes do not take effect until the second step of the printing process.

The FSLETTER ATTR window is partitioned into a series of frames. Each frame contains the field attributes for a particular field. The attribute frames are displayed in the order in which the fields appear in the document. If the same field name is used more than once in the document, each occurrence has a separate attribute frame. Display 6.3 on page 79 shows a sample field attribute frame.

Display 6.3 Sample Field Attribute Frame



Use the FORWARD and BACKWARD commands to move among the attribute frames for the document.

The attributes for each frame are choice groups. To turn on an attribute for a field, move the cursor to the term in the attribute frame that corresponds to the desired attribute; then press ENTER. The term is highlighted in reverse video to show that it is turned on. To turn off a currently selected attribute, move the cursor to the highlighted term and press ENTER. The highlighting is removed to show that the attribute is turned off.

Field Attribute Descriptions

The following attributes can be assigned in the FSLETTER ATTR window. All of the attributes except Right Justify take effect when you enter the second phase of printing

a document (see "Printing Documents" on page 84). The Right Justify attribute takes effect after you enter a value into the field.

DEAR
: modifies the field value by excluding all but the first and last elements of the value.

To assign this attribute, move the cursor to the DEAR item and press ENTER. The item is highlighted in reverse video. To turn off the attribute, move the cursor to the highlighted item and press ENTER.

This attribute converts names in the form *title first-name last-name* into *title last-name* for use in the salutations of letters. For example, if the name **`Mr. John E. Doe`** is entered in a field to which the DEAR attribute is assigned, the field value is converted to **`Mr. Doe`** when the document is printed.

Note: The DEAR attribute does not automatically provide the *Dear* element of the salutation. You must provide that in text:

```
Dear &NAME_____________________:
```

△

The DEAR attribute can be used in conjunction with the LASTNAME attribute. If the LASTNAME and DEAR attributes are both assigned, the LASTNAME attribute is processed first so that the field value is in the proper order for the DEAR attribute.

DELETE
: deletes the line that contains the field if the line is blank when the letter is printed.

To assign this attribute, move the cursor to the DELETE item and press ENTER. The item is highlighted in reverse video. To turn off the attribute, move the cursor to the highlighted item and press ENTER.

Assign the DELETE attribute if it is possible that one or more observations will not have a value for the field or if the field might not be needed for a particular document. For example, some letters require more address lines than others. Suppose your document provides three fields for the address. Some letters require only two address lines, so you can assign the DELETE attribute to the last address field. If the field is blank, the FSLETTER procedure deletes the line allotted for the field when the letter is printed. If you do not assign the DELETE attribute to the field, a blank line appears in its place when the letter is printed.

FLOW AFTER
: removes extra spaces beginning with the first character that follows the field and ending with the first blank line after the field. If the field ends the line and paragraph, then any following blank lines between the line that contains the field and the next nonblank line are removed.

To assign this attribute, move the cursor to the FLOW AFTER item and press ENTER. The item is highlighted in reverse video. To turn off the attribute, move the cursor to the highlighted item and press ENTER.

FLOW BEFORE
: takes out extra spaces beginning after the last nonblank character that precedes the field and ending with the first blank line after the field. If the field starts the line, then any preceding blank lines between the line that contains the field and the previous nonblank line are removed.

To assign this attribute, move the cursor to the FLOW BEFORE item and press ENTER. The item is highlighted in reverse video. To turn off the attribute, move the cursor to the highlighted item and press ENTER.

FLOW LINE
: takes out extra spaces beginning after the last nonblank character that precedes the field and ending with the end of the line that contains the field.

To assign this attribute, move the cursor to the FLOW LINE item and press ENTER. The item is highlighted in reverse video. To turn off the attribute, move the cursor to the highlighted item and press ENTER.

LASTNAME
modifies the field value by moving the first element of the value that is followed by a comma to the end of the value.

To assign this attribute, move the cursor to the LASTNAME item and press ENTER. The item is highlighted in reverse video. To turn off the attribute, move the cursor to the highlighted item and press ENTER.

This attribute is used with fields that contain names in the form *last-name, first-name*. Since records are often kept by last name, this attribute is helpful when you are using a SAS data set to fill in field values. For example, if the LASTNAME attribute is assigned to a field, and the value entered in the field is **`Doe, Mr. John E.`**, then the value printed in the document is **`Mr. John E. Doe`**.

Note: The LASTNAME attribute has no effect if the field value does not contain a comma. Thus, you can still enter values in the normal order in fields to which the LASTNAME attribute has been assigned. △

REQUIRED
indicates that the field must be filled in before you can leave the first step of the printing process.

To assign this attribute, move the cursor to the REQUIRED item and press ENTER. The item is highlighted in reverse video. To turn off the attribute, move the cursor to the highlighted item and press ENTER.

You must use the CANCEL command to end the first printing step without supplying a value for a required field.

RIGHT JUSTIFY
indicates that the field value should be right-aligned in the space that is reserved for the field. By default, values for numeric variables are right-aligned. Values for character variables are left-aligned, unless the character variable has the $CHAR format or informat and the value has leading blanks (leading blanks are retained with this format).

To assign this attribute, move the cursor to the RIGHT JUSTIFY item and press ENTER. The item is highlighted in reverse video. To turn off the attribute, move the cursor to the highlighted item and press ENTER.

SAS variable aliases
specifies the name of the variable from a SAS data set that is used to fill in the field. You can assign up to two aliases for the field name. The field name is automatically assigned as the first alias (unless the field name is a SAS automatic variable).

FSLETTER ATTR Window Commands

In addition to the global commands that are described in Chapter 9, “SAS/FSP Software Global Commands,” on page 143, you can use the following commands in the FSLETTER ATTR window:

Scrolling

BACKWARD

BOTTOM

FORWARD

HSCROLL HALF | PAGE | *n*

LEFT <HALF | PAGE | MAX | *n*>

RIGHT <HALF | PAGE | MAX | *n*>

TOP

Searching

BFIND <*character-string* | *'character-string*'>

BLOCATE <*value* | *'value*'>

BLOCATE: <*partial-value* | *'partial-value*'>

FIND <*character-string* | *'character-string*'>

KEYFIELD <*field-name* >

LOCATE <*value* | *'value*'>

LOCATE: <*partial-value* | *'partial-value*'>

RFIND

RLOCATE

Closing the Window

CANCEL

END

Command Descriptions

Here are descriptions of the FSLETTER ATTR window commands:

BACKWARD
displays the attribute frame for the previous field.

BFIND <*character-string* | *'character-string*'>
searches backward for the specified character string in the field that was identified in the KEYFIELD command. If the search string contains special characters, embedded blanks, or lowercase characters, enclose it in quotes.

Unlike the BLOCATE command, the BFIND command looks for the string anywhere in the field, not just at the beginning.

To search forward from the cursor position toward the bottom of the window rather than backward toward the top, use the FIND command.

To search backward for the character string specified in a previous BFIND or FIND command, use the BFIND command without arguments.

BLOCATE <*value* | *'value*'>
BLOCATE: <*partial-value* | *'partial-value*'>
searches backward for the specified value in the field that was identified in the KEYFIELD command. If the search string contains special characters, embedded blanks, or lowercase characters, enclose it in quotes.

The BLOCATE command searches only for an exact match. To specify only the first part of a value, use the BLOCATE: command. To search for a match in any part of the value instead of just at the beginning, use the BFIND command.

To find the next occurrence of the specified value, use the RLOCATE command. To search forward from the cursor position toward the bottom of the window rather than backward toward the top, use the LOCATE command.

BOTTOM
displays the attribute frame for the last field in the document.

CANCEL
closes the FSLETTER ATTR window without recording any changes to the previous field attribute settings.

END
closes the FSLETTER ATTR window and records any changes that you made to field attribute settings while the window was open.

FIND <*character-string* | *'character-string*'>
searches for the specified character string in the field that was identified in the KEYFIELD command. If the search string contains special characters, embedded blanks, or lowercase characters, enclose it in quotes.

Unlike the LOCATE command, the FIND command looks for the string anywhere in the field, not just at the beginning.

To find the next occurrence of the specified string, use the RFIND command. To search backward from the cursor position toward the top of the window rather than forward toward the end, use the BFIND command.

FORWARD
displays the attribute frame for the next field.

HSCROLL HALF | PAGE | *n*
specifies the default horizontal scroll amount for the LEFT and RIGHT commands. Specify one of the following scroll amounts:

HALF	scroll by half the window width.
PAGE	scroll by the full window width.
n	scroll by the specified number of columns.

KEYFIELD <*field-name*>
identifies the field that is searched when you next issue a FIND, LOCATE, BFIND, or BLOCATE command. Indicate the field you want to search either by supplying the field name as an argument in the KEYFIELD command or by typing KEYFIELD on the command line, positioning the cursor on the desired field, and pressing ENTER.

For example, to search for all the entries named RENEWAL in the **Name** field of the Explorer window, you must first identify **Name** as the field you want to search. Type KEYFIELD on the command line, position the cursor anywhere on the **Name** field, and press ENTER. Then issue the following command to locate the first entry in the catalog that has the name RENEWAL:

```
locate renewal
```

To display the current key field value on the window's message line, use the KEYFIELD command without arguments.

LEFT <HALF | PAGE | MAX | *n*>
scrolls the window horizontally by the specified amount. The following scroll amounts can be specified:

HALF	scrolls by half the window width.
PAGE	scrolls by the full window width.
MAX	scrolls to the left margin of the window.
n	scrolls by the specified number of columns.

If you do not explicitly specify a scrolling increment, the default increment is the amount specified in the HSCROLL command. The default HSCROLL amount is HALF.

LOCATE <*value* | '*value*'>
LOCATE: <*partial-value* | '*partial-value*'>
searches for the next occurrence of the specified value in the field that was identified in the KEYFIELD command. If the search string contains special characters, embedded blanks, or lowercase characters, enclose it in quotes.

The LOCATE command searches only for an exact match. To specify only the first part of a value, use the LOCATE: command. To search for a match in any part of the value instead of just at the beginning, use the FIND command.

To find the next occurrence of the specified value, use the RLOCATE command. To search backward from the cursor position toward the top of the window rather than forward toward the end, use the BLOCATE command.

RFIND
repeats the most recent FIND command, locating the next occurrence of the character string.

RIGHT <HALF | PAGE | MAX | *n*>
scrolls the window horizontally by the specified amount. The following scroll amounts can be specified:

HALF	scrolls by half the window width.
PAGE	scrolls by the full window width.
MAX	scrolls to the right margin of the window.
n	scrolls by the specified number of columns.

If you do not explicitly specify a scrolling increment, the default increment is the amount specified in the HSCROLL command. The default HSCROLL amount is HALF.

RLOCATE
repeats the most recent LOCATE or BLOCATE command, locating the next occurrence of the value.

TOP
displays the attribute frame for the first field in the document.

Printing Documents

The FSLETTER procedure provides two ways to print documents:

- noninteractively. When you use the DATA= option in the PROC FSLETTER statement, or the ALL or DATA= options in the SEND command, the procedure prints copies of a document for each observation in a SAS data set. In this case, no windows are opened.
- interactively. When you use the SEND command without the ALL or DATA= options, the procedure opens the FSLETTER SEND window. In this case, you can enter and edit field values manually, and you can make final editing changes before the document is printed.

Printing documents from the FSLETTER SEND window is a two-step process:

1. Fill in the fields in the document.
2. Perform final editing and release the document.

The rest of this section explains how the FSLETTER SEND window is used to prepare documents for printing.

Step 1: Filling Fields

Note: If your document does not contain fields, go directly to step 2. △

During this step, you can enter text only in the fields; all other text is protected. As you fill in each field, press the ENTER key to position the cursor at the beginning of the next field. If you fill a field, the cursor moves automatically to the next field.

Note: You cannot enter an underscore as part of the field value, because the FSLETTER procedure uses underscores as field pad characters. If you type an underscore as part of the field value, it is removed when the FSLETTER procedure processes the field value (when you press the ENTER key). This restriction is not applicable to fields that are filled noninteractively (from a SAS data set). △

If a data set is open when you open the FSLETTER SEND window, the fields in the document are automatically filled using the values from the data set. If the data set was opened by the DATA command, the values come from the first observation in the data set. If the data set is open because the FSLETTER session was invoked from the FSBROWSE or FSEDIT windows, then the values come from the observation that was displayed in the FSBROWSE or FSEDIT window when you entered the FSLETTER session.

The FSLETTER SEND window uses the SAS text editor. In the first step of sending a document, you can use any of the text editor commands described in the online Help for Base SAS software as well as the global commands described in Chapter 9, "SAS/FSP Software Global Commands," on page 143, plus the following commands:

CANCEL
: closes the FSLETTER SEND window without printing the document and returns you to the previous FSLETTER window, or to the Explorer window if no other FSLETTER windows are open. Use the CANCEL command if you decide after opening the FSLETTER SEND window that you do not want to print the document.

END
: enters the final editing step of the printing process.

Step 2: Final Editing

At the beginning of this step, any field attributes that you assigned earlier take effect. For example, fields that have the attributes LASTNAME and DEAR are rearranged, any flowing that you specified is done, and unused underscores are deleted from fields. All text in the FSLETTER SEND window is unprotected; you can type over any part of the document—field values as well as text.

In this step you can make any final editing changes that are required for the document. Changes that you make in the FSLETTER SEND window are not recorded in the stored copy of the document.

When you are ready to print the document, issue the END command again. The document is sent, and you return to the previous FSLETTER window, or to the Explorer window if no other FSLETTER windows are open.

By default, the document is sent to the destination that is specified in the form associated with the document. To route the output to an external file, use the PRTFILE= option in the PROC FSLETTER statement, the PRTFILE command in the FSLETTER window, or the FILE= option in the SEND command.

Note: If you want to route output to an external file, you must select the output file before you enter the FSLETTER SEND window. Once you open the FSLETTER SEND

window, you cannot use the PRTFILE command to change the output destination for the current document. △

In the second step of sending a document, you can use any of the commands described in step 1. However, in step 2 the END command performs different actions:

END
closes the FSLETTER SEND window, sends the document to the specified printer or file, and returns to the previous FSLETTER window, or to the Explorer window if no other FSLETTER windows are open.

Setting Text Editor Parameters

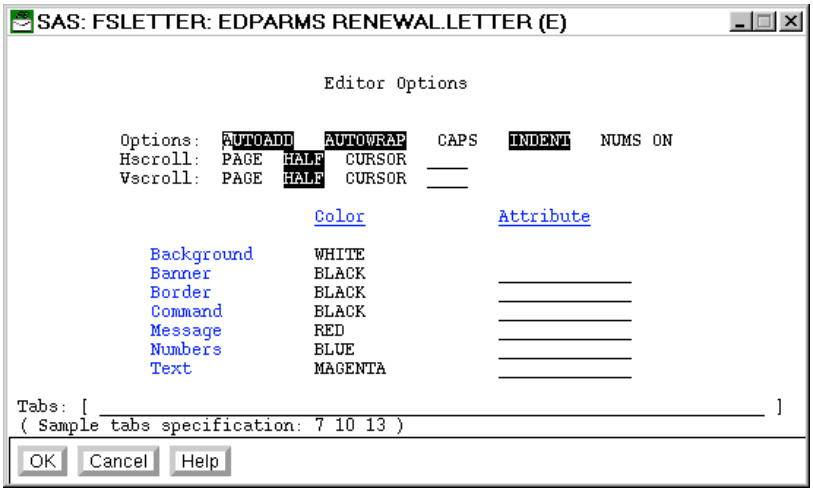
The working environment for entering, editing, or browsing text in the FSLETTER window is provided by the SAS text editor. The appearance and behavior of the SAS text editor in the FSLETTER window is determined by the settings of parameters that are associated with the document that is being edited.

To review or customize the text editor environment for a document that you are currently composing or for a previously created document that you are now editing, issue the following command while the document is displayed in the FSLETTER window:

```
edparms
```

This command opens the EDPARMS window to display the text editor parameters that are in effect for the current document. Display 6.4 on page 86 shows the EDPARMS window for a typical FSLETTER document.

Display 6.4 EDPARMS Window Showing Parameter Settings for an FSLETTER Document



You can change the parameter settings in the EDPARMS window. Some of the parameter settings in the EDPARMS window are choice groups. To turn on these parameters, move the cursor to the corresponding term in the EDPARMS window and press ENTER. The term is highlighted in reverse video to show that the parameter is turned on. To turn off a parameter, position the cursor on the highlighted term and press ENTER. The highlighting is removed to show that the parameter is turned off.

Other parameter settings are fields in which you enter the desired parameter values. The following section describes the parameters that are available in the EDPARMS window.

The editor parameters that are displayed by the EDPARMS command are associated with the displayed document only. Any parameter changes that you make are applicable only to the current LETTER entry. The changes take effect when you close the EDPARMS window.

Parameter Descriptions

The following text editor parameters are used in the FSLETTER window. The default settings of these parameters are determined by the values in the FSLETTER.EDPARMS entry when the document is created. Refer to "Creating Default Parameter Settings" on page 90 for details.

AUTOADD
controls the automatic insertion of new lines as you scroll forward past existing text. If the AUTOADD parameter is off, then you must issue a specific command, such as the I (insert) line command, to insert new lines of text in the FSLETTER window.

AUTOWRAP
controls whether a word that will not fit on the current line is automatically moved (wrapped) to the next line.

If the AUTOWRAP parameter is turned on, then you can enter text continuously, without moving the cursor to the next line. You can also use the INCLUDE command to bring a file that has a longer line length into the text editor without truncating any text. When the INDENT parameter is also turned on, you can wrap text that is indented.

Note: This parameter is not supported by some display devices. The AUTOWRAP parameter is ignored if the display device does not support the feature. (In particular, most terminals for mainframe hosts do not support autowrapping.) △

CAPS
controls whether all text that is subsequently entered or modified is converted to uppercase characters when you press ENTER or a function key, or when you move the cursor from the line you are working on. If the CAPS parameter is turned on, character strings for the FIND and CHANGE command are also converted to uppercase. If you do not want lowercase characters in the string to be converted to uppercase, enclose your character string in quotes. When the CAPS parameter is turned off, all text is left as entered.

INDENT
controls whether indention at the left margin remains when text is flowed or automatically wrapped to the next line.

NUMS ON
controls line numbering for the FSLETTER window. If your window already contains text when you turn on line numbering, all text is shifted to the right, and the line numbers appear on the left side of the window. When line numbers are displayed, you can use them to issue text editor line commands. (See the online Help for Base SAS software for details about text editor line commands.)

Hscroll
controls the default horizontal scrolling amount for the LEFT and RIGHT commands. Specify one of the following amounts:

HALF	scroll by half the current window width.
PAGE	scroll by the entire current window width.

CURSOR	scroll to the current cursor position.
n	scroll by the specified number of columns.

Vscroll
controls the default vertical scrolling amounts for the BACKWARD and FORWARD commands. Specify one of the following amounts:

HALF	scroll by half the current window height.
PAGE	scroll by the entire current window height.
CURSOR	scroll to the current cursor position.
n	scroll by the specified number of lines.

Color and Attribute
control the color and highlighting attributes of the following areas of the FSLETTER window:

Background	controls the background color of the FSLETTER window.
Banner	controls the color of the **`Command===>`** at the left of the command line. *Note:* This parameter has no effect if a menu bar is displayed in place of a command line. △
Border	controls the color of the window border in character-based display environments. *Note:* This parameter has no effect in graphical windowing environments. △
Command	controls the color of the text that users type on the command line. *Note:* This parameter has no effect if a menu bar is displayed in place of a command line. △
Message	controls the color of text that is displayed in the window's message line.
Numbers	controls the color of line numbers if they are turned on in the FSLETTER window.
Text	controls the color of any new text that you type in the FSLETTER window. *Note:* Changing this parameter does not change the color of any text that has already been entered in the FSLETTER window. △

The following values are valid in the **`Color`** fields:

BLUE	PINK	CYAN	WHITE
RED	GREEN	YELLOW	BLACK
MAGENTA	GRAY	BROWN	ORANGE

If a specified color is not available on a user's device, the procedure substitutes the available color that most closely matches the specified color. Some devices do

not allow the background color to be changed; for these devices, the background color parameter is ignored.

The following values are valid in the **Attribute** fields. (You can abbreviate the values by entering only the first letter; the procedure fills in the complete value when you press ENTER):

HIGHLIGHT	high intensity
BLINKING	blinking
UNDERLINE	underlined
REVERSE	reverse video

If a parameter specifies a highlighting attribute that is not available on the device, the parameter is ignored. You cannot specify a highlighting attribute for the background.

Tabs
controls the default tab stops for the text editor.

Specify the tab stop for each column explicitly in the field provided. For example, the following values set tab stops at columns 1 (default), 25, 40, and 55:

```
25 40 55
```

Move your cursor to a tab stop by pressing your keyboard's TAB key.

EDPARMS Window Commands

In addition to the global commands described in Chapter 9, "SAS/FSP Software Global Commands," on page 143, you can use the following commands in the EDPARMS window:

CANCEL
closes the EDPARMS window without recording any changes made in the window and returns you to the window from which the EDPARMS command was issued. Use this command when you decide not to change the current parameter settings after you have opened the EDPARMS window.

END
closes the EDPARMS window, records any changes that you made to the parameter settings in the LETTER entry, and returns you to the window from which the EDPARMS window was opened.

Using Text Editor Commands

Another way to change the text editor environment while you are editing a particular document is to use global text editor commands. There is a corresponding text editor command for each of the parameters in the EDPARMS window.

Parameter	Text Editor Command
AUTOADD	AUTOADD <ON|OFF>
AUTOWRAP	AUTOWRAP <ON|OFF>
CAPS	CAPS <ON|OFF>

Parameter	Text Editor Command
INDENT	INDENT <ON|OFF>
NUMS ON	NUMS <ON|OFF>
Hscroll	HSCROLL HALF|PAGE|CURSOR|MAX|*n*
Vscroll	VSCROLL HALF|PAGE|CURSOR|MAX|*n*
Colors and Attributes	COLOR *area color* <*highlight*>
Tabs	TABS *n*

If you open the EDPARMS window after issuing one of these commands, you see that the parameter setting in the EDPARMS window changes to reflect the parameter setting that you specified in the command.

Creating Default Parameter Settings

Whenever you create a new document, the initial settings for the text editor parameters are copied from a catalog entry named FSLETTER.EDPARMS. The default FSLETTER.EDPARMS entry is stored in the SASHELP.FSP system catalog. You can use the FSLETTER procedure to customize the default parameter entry for a catalog.

To create a custom FSLETTER.EDPARMS entry, issue the following command in an FSLETTER window:

```
edit fsletter.edparms
```

If the current catalog does not already contain an FSLETTER.EDPARMS entry, then a copy of the FSLETTER.EDPARMS entry from the SASHELP.FSP system catalog is created in the current catalog and is then opened for editing.

This command opens an EDPARMS window like the window in Display 6.4 on page 86 (except that the window title does not include the word EDPARMS). However, the parameter settings in this window have no effect on existing documents. Instead, these parameter settings become the default settings for any new documents that are created in the current catalog. You can change the parameters in FSLETTER.EDPARMS whenever you want without affecting existing LETTER entries.

You may find it convenient to create different FSLETTER.EDPARMS entries for different catalogs. If you keep letters in one catalog, reports in another, and questionnaires in another, you can create a different FSLETTER.EDPARMS entry for each catalog to store appropriate parameter settings for each kind of document.

When you create a new document, the FSLETTER procedure locates the default text editor parameter values by looking for the FSLETTER.EDPARMS entry first in the current catalog, then in your personal PROFILE catalog (SASUSER.PROFILE, or WORK.PROFILE if the SASUSER library is not allocated). If the entry does not exist in either of those catalogs, then the FSLETTER procedure uses the default FSLETTER.EDPARMS entry in the SASHELP.FSP system catalog.

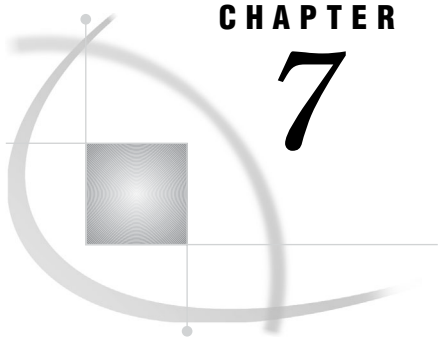
Thus, you may also want to create a custom FSLETTER.EDPARMS entry in your personal PROFILE catalog. That entry then provides default parameter values for any catalog that does not have its own FSLETTER.EDPARMS entry.

Copying Parameter Entries to New Catalogs

You do not have to create a new FSLETTER.EDPARMS entry each time you create a new catalog. If you do not want to use the default text editor parameters in the SASHELP.FSP catalog or in your personal PROFILE catalog, you can copy your custom FSLETTER.EDPARMS entry from one catalog to another.

Suppose that you have defined an FSLETTER.EDPARMS entry in an existing catalog named MASTER.LETTERS and that you have created a new catalog named MASTER.LETTERS2. You can use drag-and-drop functionality in the Explorer window to copy the FSLETTER.EDPARMS entry from the old catalog into the new catalog, or you can issue the following command in the Explorer window:

```
copyitem master.letters.fsletter.edparms
   master.letters2.fsletter.edparms
```



CHAPTER 7

The FSVIEW Procedure

Overview

The FSVIEW procedure enables you to browse or edit a SAS data set, displaying the data set as a table of rows and columns. You can also use it to create a new SAS data set.

The procedure provides tools for customizing an FSVIEW application. For example, you can redesign the display by changing the size, position, and colors of the FSVIEW window. You can also add computed variables, which display values that are calculated from other variables in the data set.

You can also open the FSVIEW window by issuing an FSVIEW command from any SAS System command line.

CAUTION:
The FSVIEW procedure edits a SAS data set in place. The FSVIEW procedure does not leave an unedited copy of the original. If you need to preserve the original data, be sure to make a copy of the data set before you begin editing. △

Note: Partially displayed data values are automatically protected in an FSVIEW window, and cannot be edited. △

FSVIEW Procedure Syntax

Restriction: Do not use any of the other statements when you use the NEW= option in the PROC FSVIEW statement. (The ID and VAR statements cause errors; the FORMAT, INFORMAT, and WHERE statements are ignored.)

Reminder: When you use the FORMULA= option with the PROC FSVIEW statement to load an existing formula entry, the features that are defined in the entry take precedence over the features that are specified in the FORMAT, ID, INFORMAT, and VAR statements.

PROC FSVIEW <DATA=*data-set*> | <NEW=*data-set* <LIKE=*data-set*>>
<FORMULA=*SAS-catalog*<.*formula-entry*>>
<*options*>;

FORMAT *variable-list format* <... *variable-list-n format-n*>;

ID *variable* <... *variable-n*>;

INFORMAT *variable-list informat* <... *variable-list-n informat-n*>;

VAR *variable* <... *variable-n*>;

WHERE *expression*;

The PROC FSVIEW statement is required. The other statements are optional and are used as follows:

To do this	Use this statement
Associate formats with variables in the input data set	FORMAT
Specify the variable or variables that are used to identify observations instead of using observation numbers	ID
Associate informats with variables in the input data set	INFORMAT
Select which variables are available to the procedure	VAR
Specify a condition or set of conditions that observations in the input data set must meet in order to be processed	WHERE

PROC FSVIEW Statement

Initiates the FSVIEW procedure.

Requirement: The FSVIEW procedure must have an input data set. By default, the procedure uses the most recently created data set as its input data set. You can use the DATA= option in the PROC FSVIEW statement to select a particular data set. If you do not specify a data set and none has previously been created in the current SAS session, the procedure terminates with an error message.

Tip: Use the FORMULA= option to identify a FORMULA entry in which to store formulas for computed variables and other custom features of the FSVIEW session.

PROC FSVIEW <DATA=*data-set*> | <NEW=*data-set* <LIKE=*data-set*>>
<FORMULA=*SAS-catalog*<.*formula-entry*>>
<AUTOADD>
<BROWSEONLY>
<DEBUG>
<EDIT | MODIFY>
<NOADD>
<NOBORDER>
<NODELETE>

Options

AUTOADD
turns on the autoadd feature. A new observation is displayed when the FSVIEW window is opened. Another new observation is added automatically whenever you enter values in the previous new observation and press ENTER.

The AUTOADD option is ignored if you also specify the BROWSEONLY or NOADD option with the PROC FSVIEW statement.

BROWSEONLY
BRONLY
prevents editing of data sets during the FSVIEW session. This option disables the MODIFY command so that data sets that are opened for browsing cannot be switched to editing. It also disables the EDIT command so that additional FSVIEW windows cannot be opened for editing.

DATA=*data-set*<(*data-set-options*)>
names the SAS data set to edit or browse. By default, the FSVIEW procedure displays the most recently created data set.

If you specify both the DATA= option and the NEW= option in the same PROC FSVIEW statement, the DATA= option is ignored.

You can add data set options following the data set name. The options must be enclosed in parentheses. The FIRSTOBS= and OBS= options are ignored; all other data set options are valid. Refer to *SAS Language Reference: Dictionary* for a list and descriptions of data set options.

DEBUG
turns on the SAS Component Language (SCL) source-level debugger. The debugger is useful for finding and correcting errors in formulas. See *SAS Component Language: Reference* for information on using the debugger.

EDIT
MODIFY
opens the initial FSVIEW window for editing the data set. By default, the initial FSVIEW window is opened for browsing.

This option is ignored if the BROWSEONLY option is also used. If the data set cannot be opened for editing, it is opened for browsing instead.

FORMULA=*SAS-catalog*<*.formula-entry*>
names a SAS catalog or specific FORMULA entry that contains formulas for computed variables and other FSVIEW session parameters to be associated with the data set, or in which the procedure can store formulas and parameter settings that are defined during the current FSVIEW session.

The general form of the *SAS-catalog* value is

<*libref.*>*catalog-name*

If the specified catalog does not already exist, it is created.

If you specify only a one-level name, it is treated as a catalog name in the default library, WORK. You must use a two-level catalog name if you want to specify a FORMULA entry name.

The general form of the *formula-entry* value is

entry-name<.FORMULA>

If you specify a two-level entry name with anything other than FORMULA as the second level (entry type), the specified type is ignored; FORMULA is used instead.

When an entry name is provided, the procedure attempts to load the specified entry when the initial FSVIEW window is opened. If the entry is not found, the FSVIEW session is initiated with default FSVIEW window characteristics. A FORMULA entry that has the specified name is created when you end the FSVIEW session or when you use the SAVE FORMULA command.

Note: If any variable names that are specified in the FORMULA entry do not match those in the data set to be displayed, then the FORMULA entry is not loaded and an error message is displayed. △

LIKE=*data-set*<(*data-set-options*)>
names an existing SAS data set whose structure is copied when a new SAS data set is created. (This option must be used in conjunction with the NEW= option.) When the FSVIEW NEW window is opened, the variable names and attributes of the specified data set are displayed.

You can add data set options following the data set name. The options must be enclosed in parentheses. Refer to *SAS Language Reference: Dictionary* for a list and descriptions of data set options.

MODIFY
See the EDIT option.

NEW=*data-set*<(*data-set-options*)>
creates a new SAS data set. If a data set that has the specified name already exists, the FSVIEW procedure terminates with an error message.

When this option is used, the procedure begins by opening the FSVIEW NEW window, in which the names and attributes of the variables in the new data set are defined. Use the LIKE= option in conjunction with the NEW= option to initialize the FSVIEW NEW window with the variable names and attributes of an existing data

set. After the structure of the new data set is defined, the FSVIEW window is opened so that observations can be added. For details, see "Creating New SAS Data Sets" on page 128.

If you specify both the NEW= option and the DATA= option in the same PROC FSVIEW statement, the DATA= option is ignored.

You can add data set options following the data set name. The options must be enclosed in parentheses. Refer to *SAS Language Reference: Dictionary* for a list and descriptions of data set options.

NOADD
prevents the addition of observations to any data set that is edited during the current FSVIEW session. This option disables the AUTOADD and DUP commands so that no new observations can be added.

NOBORDER
suppresses the sides and bottom of the FSVIEW window's border in a character-based display environment.

Note: This option is ignored in graphical windowing environments. △

When this option is used in a supported display environment, values can appear in the columns and row that the border normally occupies.

When the NOBORDER option is used in conjunction with the FORMULA= option, the window size that is specified in the FORMULA entry is ignored. The FSVIEW window always occupies the maximum possible number of rows and columns when the NOBORDER option is specified.

NODELETE
NODEL
prevents the deletion of observations from any data set that is edited during the current FSVIEW session. This option disables the DELETE command so that observations cannot be deleted.

FORMAT Statement

Associates formats with variables in the input data set. *Formats* are patterns that the SAS System uses to determine how variable values are displayed. The formats can be either SAS formats or custom formats that you have defined with the FORMAT procedure.

Reminder: The FORMAT statement is ignored when you use the NEW= option in the PROC FSVIEW statement.

Note: If you load an existing FORMULA entry, any formats that are recorded in the FORMULA entry take precedence over the formats that are specified in the FORMAT statement.

FORMAT *variable-list format* <*... variable-list-n format-n*>;

Arguments

At least one pair of the following arguments is required:

variable-list
consists of one or more variable names from the input data set.

format
is the SAS format or user-defined format to associate with the specified variable or variables.

Using the FORMAT Statement

You can use a single FORMAT statement to assign the same format to several variables or to assign different formats to different variables. You can use any number of FORMAT statements with each PROC FSVIEW statement.

Formats that are specified in a FORMAT statement take precedence over formats that are defined in the data set itself. Formats that are assigned in a FORMAT statement remain in effect only for the duration of the procedure; the FORMAT statement does not affect any format assignments that are stored in the data set. You can use the FORMAT command in the FSVIEW window to override formats that were specified in the FORMAT statement.

Format widths determine the width of the columns for the associated variables in the FSVIEW display.

Be aware that the format you assign a variable affects the informats you can assign with the INFORMAT statement. For example, suppose the data set that is displayed by the FSVIEW procedure contains a variable AMOUNT that is assigned the format DOLLAR10.2 but an informat of 10.2. Because of the format, values in the AMOUNT column are displayed with commas and a leading dollar sign, so the value 1250 would be displayed as $1,250.00. However, if you edit this value (for example, changing it to $1,150.00) and press ENTER, an error condition occurs. The 10.2 informat does not permit the dollar sign ($) or comma characters in entered values. An appropriate informat for this variable is COMMA., which does permit these characters.

Refer to *SAS Language Reference: Dictionary* for more detailed information on formats and on the FORMAT statement. See the description of the FORMAT procedure in *Base SAS Procedures Guide* for information on defining your own formats with the FORMAT procedure.

ID Statement

Specifies one or more variables whose values identify observations in the FSVIEW window.

Restriction: If you use the ID statement in conjunction with a PROC FSVIEW statement that includes the NEW= option, the FSVIEW procedure terminates with an error message.

Note: If you load an existing FORMULA entry, the ID variables that are recorded in the FORMULA entry take precedence over the ID variables that are specified in the ID statement.

ID *variable* <*... variable-n*>;

Argument

variable is the name of a variable from the displayed data set.

Using the ID Statement

By default, the FSVIEW procedure uses observation numbers to identify observations. Observation numbers are not displayed when ID variables are used. The values of the ID variables appear at the beginning of each row and remain on the left side of the window when the other variable columns are scrolled horizontally.

The FSVIEW procedure does not show more than the first 53 characters of ID variables. If the ID variable column exceeds this width, it is truncated.

If you want to know an observation's number when it is identified by an ID variable, use the OBS command. See "FSVIEW Window Commands" on page 108 for details.

Use the COLOR command to select the color for ID variables and for the ID variable column heading. See "FSVIEW Window Commands" on page 108 for details.

INFORMAT Statement

Associates informats with variables in the input data set. *Informats* are patterns that the SAS System uses to determine how values that are entered in variable columns are interpreted. The informats can be either SAS informats or custom informats that you have defined with the FORMAT procedure.

Reminder: INFORMAT statements are ignored if you use the NEW= option in the PROC FSVIEW statement.

Note: If you load an existing FORMULA entry, any informats that are recorded in the FORMULA entry take precedence over the informats that are specified in the INFORMAT statement.

INFORMAT *variable-list informat* <*... variable-list-n informat-n*>;

Arguments

At least one pair of the following arguments is required:

variable-list
consists of one or more variable names from the input data set.

informat
is the SAS informat or user-defined informat to associate with the specified variable or variables.

Using the INFORMAT Statement

You can use a single INFORMAT statement to assign the same informat to several variables or to assign different informats to different variables. You can use any number of INFORMAT statements with each PROC FSVIEW statement.

Informats that are specified in an INFORMAT statement take precedence over informats that are defined in the data set itself. Informats that are assigned in an INFORMAT statement remain in effect only for the duration of the procedure; the INFORMAT statement does not affect any informat assignments that are stored in the data set. You can use the INFORMAT command in the FSVIEW window to override informats that were specified in the INFORMAT statement.

When using INFORMAT statements with the FSVIEW procedure, you should make sure the informats that you assign to variables are compatible with the formats for

those variables. That is, the output that the format produces should be valid input for the informat. Otherwise, you complicate the process of editing variable values. For example, suppose the data set that is displayed by the FSVIEW procedure contains a variable AMOUNT that is assigned the informat 10.2 and the format DOLLAR10.2. Because of the format, values in the AMOUNT column are displayed with commas and a leading dollar sign, so the value 1250 would be displayed as $1,250.00. However, if you edit this value (for example, changing it to $1,150.00) and press ENTER, an error condition occurs. The 10.2 informat does not permit the dollar sign ($) or comma characters in entered values. An appropriate informat for this variable is COMMA., which does permit these characters.

Refer to *SAS Language Reference: Dictionary* for more detailed information on informats and on the INFORMAT statement. See the description of the FORMAT procedure in *Base SAS Procedures Guide* for information on defining your own informats with the FORMAT procedure.

VAR Statement

Selects which variables to display and what order to display them in.

Restriction: The FSVIEW procedure terminates with an error message if you use the VAR statement in conjunction with a PROC FSVIEW statement that includes the NEW= option.

Note: If you load an existing FORMULA entry, the variables and variable order that are recorded in the FORMULA entry take precedence over the list and order of variables that are specified in the VAR statement.

Tip: When more than one variable name is supplied, any valid form of variable list can be used. See *SAS Language Reference: Concepts* for details about variable lists.

VAR *variable* <*... variable-n*>;

Argument

variable is the name of a variable from the displayed data set.

Using the VAR Statement

By default, the FSVIEW procedure displays all the variables from the data set in the FSVIEW window and arranges the variable columns in the order in which the variables appear in the data set (except for any variables that are used as ID variables). You can use the VAR statement to control which variables appear in the FSVIEW window and in what order they appear. Columns of variable values appear in the order that is specified in the VAR statement. The number of columns that are visible at any given time depends on the width of the variable values and on the current width of the FSVIEW window.

If the same variable name appears in both the ID and VAR statements, the variable is used only as an ID variable.

During an FSVIEW session, use the SHOW and DROP commands to control the display of variables. The SHOW command can add variables to the display even if they

are not listed in the VAR statement. See "FSVIEW Window Commands" on page 108 for details.

Use the COLOR command to select the colors for displayed variables and column headings. See "FSVIEW Window Commands" on page 108 for details.

WHERE Statement

Defines criteria that observations in the input data set must meet in order to be displayed by the procedure.

Reminder: The WHERE statement is ignored if you use the NEW= option in the PROC FSVIEW statement.

WHERE *expression*;

Argument

expression is any valid WHERE expression that includes one or more of the variables in the input data set. Refer to the description of the WHERE statement in *SAS Language Reference: Dictionary* for details about the operators and operands that are valid in WHERE expressions.

Using the WHERE Statement

By default, the FSVIEW procedure displays all the observations in the data set. The WHERE statement is useful when you want to view only a subset of the observations in a SAS data set. For example, to view only observations for which the value of the variable YEAR is less than 5, follow the PROC FSVIEW statement with this statement:

```
where year<5;
```

The FSVIEW procedure displays only observations that meet the specified condition(s). Observations that do not satisfy the condition(s) are not shown and cannot be edited.

Conditions that are imposed by a WHERE statement (or, equivalently, by a WHERE= data set option) are called permanent WHERE clauses because they remain in effect for the duration of the FSVIEW session and cannot be canceled or modified while the procedure is active.

When you use a WHERE statement in conjunction with the PROC FSVIEW statement, the behavior of the FSVIEW procedure is affected in the following ways:

- The word *Subset* appears in parentheses following the FSVIEW window title to indicate that a permanent WHERE clause is in effect.
- Observation numbers might not be sequential because some observations might be excluded.
- You cannot use the SORT command to sort the displayed observations.
- You cannot scroll a particular observation to the top of the FSVIEW window by typing the corresponding observation number on the command line.

FSVIEW Command Syntax

Tip: The FSVIEW command provides an easy way to initiate an FSVIEW session from any SAS System command line.

FSVIEW <? | *data-set* <*formula-name*>>

FSVIEW Command

Initiates an FSVIEW session.

FSVIEW <? | *data-set* <*formula-name*>>

Arguments

You can specify the following arguments with the FSVIEW command:

?
opens a selection window from which you can choose the data set to be displayed by the FSVIEW procedure. The selection list in the window includes all data sets in all SAS data libraries that have been identified in the current SAS session (all data libraries that have defined librefs).

To select a data set, position the cursor on the desired data set name and press the ENTER key, or enter the desired data set name in the **Member Name** field.

data-set
specifies the data set to be displayed by the FSVIEW procedure. The general form of the argument is

<*libref.*>*data-set-name*<(*data-set-options*)>

If you omit the libref, then the default library, WORK, is assumed.

If you specify a data set that does not exist, a selection window is opened showing all available data sets. An error message in the selection window indicates that the specified data set does not exist.

If you omit this argument and do not specify **?** for a selection window, the most recently created data set (the one identified in the _LAST_= system option) is selected. If no data set has previously been created in the current SAS session, a selection window is opened showing all available data sets.

You can add a list of data set options following the data set name. The options must be enclosed in parentheses. The FIRSTOBS= and OBS= options are ignored; all other data set options are valid. Refer to *SAS Language Reference: Dictionary* for a list and descriptions of data set options.

formula-name
specifies a FORMULA entry that is associated with the FSVIEW session. The general form of the argument is

<*libref.*>*catalog-name*<.*entry-name*<.FORMULA>>

You can specify a one-, two-, three-, or four-level name:

- If a one-level name is specified, it is treated as a catalog name in the default library, WORK. If the specified catalog does not already exist in the WORK

library, it is created. The procedure does not attempt to load a FORMULA entry when the FSVIEW session is initiated. When you end the FSVIEW session, a FORMULA entry is created, using the data set name as the FORMULA entry name.

Remember that all catalogs in the WORK library are erased when you end your SAS session.

- If a two-level name is specified, it is treated as *libref.catalog-name*. If the specified catalog does not already exist in the specified library, it is created. The procedure does not attempt to load a FORMULA entry when the FSVIEW session is initiated. When you end the FSVIEW session, a FORMULA entry is created in the specified catalog, using the data set name as the FORMULA entry name. (If an entry that has that name already exists in the catalog, it is replaced without warning.)

  For example, suppose you initiate an FSVIEW session with the following command:

  ```
  fsview master.subscrib master.custom
  ```

  When you end the FSVIEW session, the FORMULA entry MASTER.CUSTOM.SUBSCRIB.FORMULA is created.

- If a three-level name is specified, it is treated as *libref.catalog-name.entry-name*. The entry type is assumed to be FORMULA. The procedure attempts to load the specified FORMULA entry when the FSVIEW session is initiated. If the specified FORMULA entry is not found, the FSVIEW session is initiated without an associated FORMULA entry. A FORMULA entry that has the specified name (and an entry type of FORMULA) is created when you end the FSVIEW session.

- If a four-level name is specified, the fourth level (entry type) should be FORMULA. Any other value for the type is ignored; FORMULA is used instead. The procedure attempts to load the specified FORMULA entry when the FSVIEW session is initiated. If the specified FORMULA entry is not found, the FSVIEW session is initiated without an associated FORMULA entry. A FORMULA entry that has the specified name is created when you end the FSVIEW session.

  If any of the variable names that are specified in the FORMULA entry do not match those in the data set to be displayed, then the FORMULA entry is not loaded, and an error message is displayed.

Using the FSVIEW Command

The FSVIEW command always opens the data set for browsing. Use the MODIFY command to edit the displayed entries.

When you end an FSVIEW session that was initiated with the FSVIEW command, you return to the window that was active when the command was issued. Thus, the command is particularly useful when you want to view the contents of a data set while another interactive windowing procedure is executing.

The FSVIEW command does not support procedure options such as BROWSEONLY and NOADD. You must use the PROC FSVIEW statement rather than the FSVIEW command if you want to invoke the procedure with these options, or if you want to establish default procedure characteristics with the FORMAT, ID, INFORMAT, VAR, or WHERE statements.

CHAPTER

8

FSVIEW Procedure Windows

Overview

You can use the FSVIEW procedure to perform the following tasks:

- browse and edit existing SAS data sets
- create new SAS data sets
- duplicate existing SAS data sets

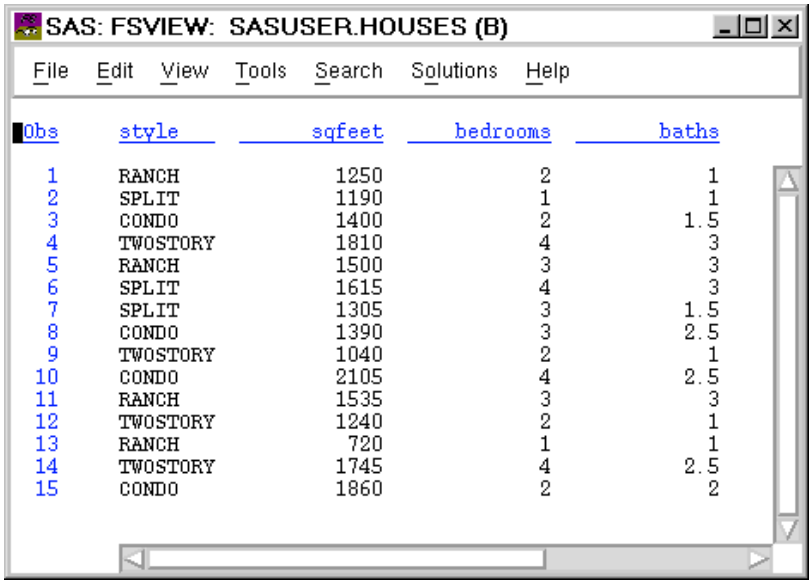
- define formulas for creating computed variables or for manipulating data set variables
- customize the FSVIEW environment by setting general parameters
- create data entry or data browsing applications.

The following sections describe the windows and commands you use to accomplish these tasks.

Browsing and Editing SAS Data Sets

In the FSVIEW procedure, observations are displayed in the FSVIEW window. Display 8.1 on page 106 identifies important features of the FSVIEW window.

Display 8.1 The FSVIEW Window



Obs	style	sqfeet	bedrooms	baths
1	RANCH	1250	2	1
2	SPLIT	1190	1	1
3	CONDO	1400	2	1.5
4	TWOSTORY	1810	4	3
5	RANCH	1500	3	3
6	SPLIT	1615	4	3
7	SPLIT	1305	3	1.5
8	CONDO	1390	3	2.5
9	TWOSTORY	1040	2	1
10	CONDO	2105	4	2.5
11	RANCH	1535	3	3
12	TWOSTORY	1240	2	1
13	RANCH	720	1	1
14	TWOSTORY	1745	4	2.5
15	CONDO	1860	2	2

By default,

- the leftmost column has the heading *Obs* and contains the observation number of the corresponding observations.

  Note: If the engine does not support observation numbers, the observation number column has the heading *Row* instead of *Obs*. △

  You can use variable values instead of observation numbers for row identifier columns.
- the other column headings are the names of the corresponding data set variables. You can assign other headings to the rows.
- the FSVIEW procedure opens the input data set for browsing. Use the MODIFY command if you want to edit the data set. (If you use the PROC FSVIEW statement to start the procedure, you can add the EDIT or MODIFY option to directly open the data set for editing.) If you use the procedure to create a new data set, the FSVIEW window is opened for editing after the structure of the new data set has been defined in the FSVIEW NEW window.

Unless you use a WHERE statement in conjunction with the PROC FSVIEW statement, all observations in the data set are available for browsing or editing. The FSVIEW procedure ignores the FIRSTOBS= and OBS= system options.

CAUTION:
The FSVIEW procedure edits a SAS data set in place. The FSVIEW procedure does not leave an unedited copy of the original. If you need to preserve a copy of the original data, be sure to copy the data set before you begin editing. △

Note: Partially displayed data values are automatically protected in an FSVIEW window, and cannot be edited. △

How the Control Level Affects Editing

The editing behavior of the FSVIEW procedure depends on which control level you select when the data set is opened. The *control level* is the degree to which the procedure can restrict access to the data set.

The FSVIEW procedure supports two levels of control:

record	locks only the observation that is currently being edited. With this control level, you can open multiple FSVIEW windows for browsing or editing the same data set. Other users can edit the same data set simultaneously.
member	locks the entire data set. No other window or user can open the data set while this control level is in effect.

By default, the FSVIEW procedure selects record-level control when it opens a SAS data set. You can specify the control level with the MODIFY command in the FSVIEW window, or by using the CNTLLEV= data set option with the data set name in the PROC FSVIEW statement, FSVIEW command, or BROWSE or EDIT command. The CNTLLEV= data set option is described in *SAS Language Reference: Dictionary*.

Editing with Record-level Control

When record-level control is selected, you must lock an observation for editing before you can change variable values in that observation. To lock an observation, move the cursor to the line for the desired observation and press ENTER. The line is highlighted to indicate that it is locked. (You can use the HI command to control which type of highlighting is used.) Once the observation is locked, you can change values in the variable columns. Scroll right or left to display additional columns. If you press ENTER on a column's value, the FSVIEW procedure marks the value as modified. If you have defined formulas for any data set variables, the values of the variables that have formulas are updated only when observations are locked.

The lock on the current observation is released when you lock a different observation for editing. In addition, the lock is released when any of the following commands cause the currently locked observation to scroll out of the window:

FORWARD	*n*
BACKWARD	AUTOADD
TOP	DUP
BOTTOM	

The lock is always released when you issue any of the following commands:

DELETE

SORT

WHERE

Editing with Member-level Control

When the MEMBER control level is selected, the FSVIEW procedure obtains exclusive control over the data set. In this case, all observations are available for editing. It is not necessary to select an individual observation before editing; simply move the cursor to the desired observation and enter the new value in the desired variable column.

Opening Multiple FSVIEW Windows

After entering the FSVIEW procedure, you can use the BROWSE, EDIT, and NEW commands to open additional FSVIEW windows to concurrently browse or edit other SAS data sets. The multiple FSVIEW windows may overlap visually, but each is completely independent of the others.

Opening additional FSVIEW windows within the current FSVIEW session by using the EDIT or BROWSE commands consumes fewer computer resources than starting additional FSVIEW sessions with the FSVIEW command.

If you invoke the FSVIEW procedure with a PROC FSVIEW statement, some of the PROC FSVIEW statement options affect the behavior of the additional FSVIEW windows:

- If you use the BROWSEONLY option, the MODIFY and EDIT commands are not permitted in any FSVIEW windows that are opened during the FSVIEW session.
- If you use the NOADD option, the AUTOADD and DUP commands are not permitted in any FSVIEW windows that are opened during the FSVIEW session.
- If you use the NODELETE option, the DELETE command is not permitted in any FSVIEW windows that are opened during the FSVIEW session.

FSVIEW Window Commands

In addition to the global commands that are discussed in Chapter 9, “SAS/FSP Software Global Commands,” on page 143, you can issue the following commands when editing a data set with the FSVIEW procedure. When browsing, you can use all the following commands except those that are identified as editing commands:

Scrolling

n

=*variable*

BACKWARD <*n* | HALF | PAGE | MAX>

BOTTOM

FORWARD <*n* | HALF | PAGE | MAX>

HSCROLL *n*

LEFT <*n* | MAX>

RIGHT <*n* | MAX>

SMOOTH <ON | OFF>

TOP

VSCROLL *n* | HALF | PAGE

Managing the Data Set

AUTOSAVE <*n*>

BROWSE *data-set* <FORMULA=*SAS-catalog*<.*formula-entry*>>

CANCEL

CREATE *data-set* <REPLACE> <*variable-list* | ALL | ?>

EDIT *data-set* <FORMULA=*SAS-catalog*<.*formula-entry*>>

END

MODIFY <RECORD|MEMBER>

NEW *data-set* <LIKE=*data-set*>

<FORMULA=*SAS-catalog*<.*formula-entry*>>

SAVE

WHERE <<ALSO> *expression*> <UNDO | CLEAR>

Editing the Data Set

(The commands in this group are valid only when the FSVIEW window is opened for editing.)

AUTOADD | ADD <ON | OFF>

CURSOR <*variable*>

DELETE | DEL <*obs* <... *obs-n*>>

DUP <*n* <*obs*>>

INITIAL <*obs* | CLEAR>

PROTECT ON | OFF <*variable* <... *variable-n*>>

SORT <ASCENDING | DESCENDING> *variable* <... <ASCENDING | DESCENDING> *variable-n*>

Managing Formulas

DEFINE <*variable-name* <$>
<(<*format*><,<*informat*><,*label*>>)><=*formula*>>

FORMULA <*formula-name*>

RESET <ALL | *variable* <... *variable-n*>>

REVIEW

SAVE FORMULA <*formula-name*>

Changing the Display

COLOR *area color* <*highlight*>

DROP <*variable* <*... variable-n*>>

FORMAT *variable-list 'format'* <*... variable-list-n 'format-n'*>

HI <ON | OFF> | <*color* <*highlight*>> | <RESET <ALL>>

INFORMAT *variable-list 'informat'* <*... variable-list-n 'informat-n'*>

MOVE *variable* | *variable-range* <*target-variable*>

RENAME *current-variable-name new-variable-name*

SETWSZ <CLEAR>

SHOW ID | VAR *variable* <*...variable-n*>

Other

KEYS

OBS

PARMS

Command Descriptions

Here are descriptions of the FSVIEW window commands:

n
scrolls the FSVIEW window vertically so that the specified observation occupies the top line. Type the desired observation number on the command line, then press ENTER. You receive an error message if you specify a value that is greater than the highest observation number.

The *n* command is still valid when ID variables are used in place of observation numbers to identify observations. However, the command is not valid when the engine that is being used to read the data set does not support access by observation number. When an engine does not support observation numbers, the column label is ROW instead of OBS. The command also is not valid while a permanent or temporary WHERE clause is in effect.

=*variable*
scrolls the FSVIEW window horizontally so that the column for the specified variable is visible.

AUTOADD <ON | OFF>
ADD <ON | OFF>
(editing command; not valid while browsing a data set)

turns the autoadd feature on or off. While the autoadd feature is on, a new observation is always displayed. If observation numbers are used in the ID column, the new observation is identified with the word NEW.

Note: The AUTOADD command is not permitted if the FSVIEW session is initiated with a PROC FSVIEW statement that includes the NOADD option. △

By default, all values in the new observation are missing. You can use the INITIAL command to store the current contents of an existing observation; the stored values are then copied into new observations when they are displayed by the autoadd feature.

By default, the cursor is positioned on the column for the leftmost variable in the FSVIEW window whenever a new observation is displayed. You can use the CURSOR command to specify the variable on which the cursor is positioned when the new observation is displayed.

The new observation is not actually added to the displayed data set until you add values and press ENTER, or until you issue any FSVIEW command other than LEFT or RIGHT. While entering values, you can use the LEFT and RIGHT commands to scroll among the variable columns without adding the new observation to the data set. After the observation is added, another new observation is displayed.

If the new observation has not been modified when you issue the END command, the observation is not added to the data set.

When the AUTOADD command is used without the ON or OFF arguments, it acts as a toggle, turning the autoadd feature on if it is currently off or off if it is currently on.

AUTOSAVE <*n*>

specifies how frequently the FSVIEW procedure automatically saves the data set. The *n* value determines how many modifications must be made before an automatic save is performed. By default, the procedure saves the data set automatically whenever 25 observations have been modified since the last save.

To check the current value of AUTOSAVE parameter, issue the AUTOSAVE command without specifying an *n* value.

You can change the default AUTOSAVE value for an FSVIEW application by changing the value in the **`Autosave value`** field in the FSVIEW Parameters window. See "Customizing the FSVIEW Environment" on page 136 for details.

Regardless of the AUTOSAVE value, you can save the data set at any time using the SAVE command.

BACKWARD <*n* | HALF | PAGE | MAX>

scrolls vertically toward the top of the window. The following scroll amounts can be specified:

n	scrolls upward by the specified number of observations.
HALF	scrolls upward by half the number of lines in the window.
PAGE	scrolls upward by the number of lines in the window.
MAX	scrolls upward until the first observation is at the top of the window.

If the scrolling increment is not explicitly specified, the window is scrolled by the amount specified in the VSCROLL parameter. The default VSCROLL amount is HALF.

BOTTOM

scrolls forward until the last observation in the data set is displayed at the bottom of the window.

You can interrupt the BOTTOM scroll operation before the last observation is reached. This feature is useful when you want to halt a scroll request while you are browsing or editing a large data set. To halt a BOTTOM scroll operation that is in progress, press the interrupt key or key combination for your system. The FSVIEW procedure displays a requestor window that gives you the options of canceling or resuming the scrolling operation.

Note: The key or key combination that is used to interrupt an active process depends on your host operating system and terminal device. For example, some systems have a key labeled BREAK or ATTENTION (or ATTN). Other systems use a combination of the CTRL key and another key. Refer to your host documentation to determine the interrupt key or key combination for your host operating system and terminal device. △

BROWSE *data-set*<(*data-set-options*)> <FORMULA=*SAS-catalog*<.*formula-entry*>>
opens another FSVIEW window and displays the specified SAS data set for browsing. The current FSVIEW window remains open, but the new FSVIEW window becomes the active window. All FSVIEW procedure browsing commands are valid in the new window (except for those that are disabled with PROC FSVIEW statement options). When you use the END command to close the new window, you return to the FSVIEW window from which the BROWSE command was issued.

The additional FSVIEW window may overlay the current one. To control where the additional window appears, use the WREGION command before issuing the BROWSE command. You can use the SWAP command to move among concurrent FSVIEW windows.

You can use the MODIFY command to change the new window from browsing to editing, unless the procedure is initiated by a PROC FSVIEW statement that includes the BROWSEONLY option.

You can add data set options following the data set name. The options must be enclosed in parentheses. The FIRSTOBS= and OBS= options are ignored; all other data set options are valid. Refer to *SAS Language Reference: Dictionary* for a list and descriptions of data set options.

You can add the FORMULA= argument to the BROWSE command to associate a FORMULA entry with the displayed data set. The syntax for the FORMULA= argument is the same as for the PROC FSVIEW statement's FORMULA= option.

CANCEL
closes the FSVIEW window and saves the data set without recording any updated information in the associated FORMULA entry. (Changes to variable values are recorded; changes to formulas and to other general parameters are not.) If only one FSVIEW window is open in the current FSVIEW session, the CANCEL command also ends the FSVIEW procedure.

COLOR *area color*<*highlight*>
sets the color and highlighting attributes of different areas of the window. You can specify the following areas:

VAR	variable value columns (except for ID variables)
VARNAME	variable names that are used as column headings (except for ID variables)
ID	ID value columns (or the observation number column if ID variables are not used)
IDNAME	ID variable names that are used as column headings (or the heading of the observation number column if ID variables are not used).

The following values are valid for the *color* argument:

BLACK	CYAN	MAGENTA	RED
BLUE	GRAY	ORANGE	WHITE
BROWN	GREEN	PINK	YELLOW

The following values are valid for the *highlight* argument:

H (high intensity)	U (underlined)
R (reverse video)	B (blinking)

In addition to the areas specific to the FSVIEW procedure, you can specify colors and attributes for the following FSVIEW window areas by using the global COLOR command:

BACKGROUND	BORDER	MESSAGE
BANNER	COMMAND	

Refer to the online Help for Base SAS software for more information on the COLOR command.

CREATE *data-set*<(*data-set-options*)> <REPLACE> <*variable-list* | ALL | ?>
creates a new SAS data set, using some or all of the variables from the data set that is currently displayed. The new data set duplicates both the structure and contents of the displayed data set. You can select which variables are included in the new data set in any of the following ways:

- by listing the desired variable names as command arguments.
- by using the ALL argument to select all variables in the displayed data set.
- by using the **?** argument to open the Select Table Variables window. In this window you can select the desired variables from a list of all available variables in the displayed data set. Refer to "Selecting Variables for FSVIEW Operations" on page 131 for more information on using the Select Table Variables window.

  Note: The Select Table Variables window is also opened if you issue a CREATE command that includes neither variable names nor the ALL or **?** arguments. By default, all variables are selected in the Select Table Variables window. △

Both computed variables and data set variables can be selected. Computed variables in the displayed data set become data set variables in the created data set.

You can add data set options following the data set name. The options must be enclosed in parentheses. Refer to *SAS Language Reference: Dictionary* for a list and descriptions of data set options.

By default, the CREATE command fails with an error message if the named data set already exists. If you want the new data set to replace an existing data set that has the same name, specify the REPLACE option. If you use the REPLACE option, it must precede the variable name arguments.

CURSOR <*variable*>
(editing command; not valid while browsing a data set)
specifies the variable column on which the cursor is positioned when a new observation is displayed. (New observations are created by the autoadd feature or by the DUP command.) By default, the cursor is positioned on the leftmost variable column in the FSVIEW window when the new observation is added.

You can specify the desired variable name as an argument for the CURSOR command, or you can type CURSOR on the command line, position the cursor on the desired variable column, and press ENTER.

DEFINE *<variable-name <$> <(<format><,<informat><,label>>)> <=formula>>*
defines a formula for a computed variable or for an existing data set variable.

When you use the DEFINE command alone or with only the *variable-name* argument, the FSVIEW Define window is opened. In this window you can enter all the information necessary to define the formula. You can also use this form of the command to open the FSVIEW Define window to edit an existing formula. See "Defining Formulas" on page 132 for more information on using the FSVIEW Define window.

If the variable name that you specify does not exist in the displayed data set, the FSVIEW procedure creates a computed variable. When creating a computed variable, the procedure assumes by default that the variable is numeric. To create a character variable, add the $ argument following the variable name if you are specifying the formula in the DEFINE command argument. (Do not use the $ argument if you want to open the FSVIEW Define window.)

You can supply the formula as an argument with the DEFINE command. In this case, the FSVIEW Define window is not opened. Separate the formula from the variable name with an equal sign (=).

Formulas are not limited to SAS expressions. They can also include any valid SAS Component Language (SCL) functions or statements except those that are valid only in SAS/AF software and the following statements:

CONTROL (except for CONTROL ENTER)

CURSOR

ERRORON

ERROROFF

PROTECT

UNPROTECT

Refer to *SAS Component Language: Reference* for details about including SCL elements in FSVIEW formulas.

You can specify a format, informat, and label for the variable in the DEFINE command. The arguments for these values must be enclosed in parentheses and must appear before the equal sign that begins the formula definition. The values must be specified in the order *format*, *informat*, *label*. You can omit any of the values as long as you include the necessary commas.

If you omit the *format* argument when defining a computed variable, the default format is BEST12. for numeric variables or $8. for character variables. If you omit the *informat* argument, the default informat is 12. for numeric variables and $8. for character variables.

You can also use the *format*, *informat*, and *label* arguments to change the format, informat, or label of an existing variable. For an existing variable, you can supply these arguments without supplying a formula. (The arguments must still be enclosed in parentheses.) For data set variables, these arguments change the format, informat, and label that are recorded in the FORMULA entry, but they do not affect any format, informat, or label for the variable that is recorded in the data set itself.

If the cursor is positioned on a variable column when the DEFINE command is issued, the column for the new computed variable appears immediately to the right of the column on which the cursor was positioned. Otherwise, the column for the new computed variable appears to the right of the displayed variable columns.

DELETE *<obs <... obs-n>>*

DEL <*obs* <*... obs-n*>>
(editing command; not valid while browsing a data set)
deletes one or more observations from the data set you are editing.

CAUTION:
The DELETE command deletes observations from the displayed data set, not just from the FSVIEW window. You cannot recover the contents of a deleted observation. △

Note: The DELETE command is not permitted if the FSVIEW session is initiated with a PROC FSVIEW statement that includes the NODELETE option. △

The behavior of the DELETE command depends on the control level at which the data set is currently opened for editing.

For member-level control
To delete a single observation, follow the DELETE command with the observation number of the observation to be deleted. You can also type DELETE (or DEL) on the command line, position the cursor on the observation to be deleted, and press ENTER.

To delete multiple observations, follow the DELETE command with a list of observation numbers. Separate the observation numbers with at least one space. For example, issue the following command to delete observations 5 and 10:

```
delete 5 10
```

To delete a range of observations, specify the first and last observation numbers of the range, separated by a dash. For example, the following command deletes all observations between 5 and 10, inclusive:

```
delete 5-10
```

For record-level control
You can delete only one observation at a time, and you must lock the observation before you can delete it. Either follow the DELETE command with the observation number for the locked observation to be deleted or type DELETE (or DEL) on the command line, position the cursor on the locked observation to be deleted, and press ENTER.

DROP <*variable* <*...variable-n*>>
excludes one or more variables from the FSVIEW window. The DROP command affects only the display, not the actual data set.

If you issue the DROP command without arguments, the Select Table Variables window is opened for you to select the variables to be dropped. Refer to "Selecting Variables for FSVIEW Operations" on page 131 for more information on using the Select Table Variables window.

To exclude a single variable, follow the DROP command with the name of the variable to be dropped. Alternatively, you can type DROP on the command line, position the cursor on the variable column to be dropped, and press ENTER.

To exclude multiple variables, follow the DROP command with a list of variable names. Separate the variable names with at least one space. For example, the following command drops the variables X and Z from the display:

```
drop x z
```

You can indicate a range of variables by specifying the first and last variables to be dropped, separated by a dash. For example, the following command drops all the variables between X and Z, inclusive:

```
drop x-z
```

Note: At least one variable, in addition to any ID variables specified, must remain displayed. △

Use the SHOW command to redisplay dropped variables.

DUP <*n* <*obs*>>
(editing command; not valid while browsing a data set)

copies the specified observation *n* times and adds the new observation(s) to the displayed data set. The FSVIEW window is automatically scrolled forward so that the new observations appear at the top of the window.

Note: The DUP command is not permitted if the FSVIEW session is initiated with a PROC FSVIEW statement that includes the NOADD option. △

You can select the observation to copy by supplying its number as the *obs* argument in the DUP command. To specify the *obs* argument, you must also specify the *n* argument (the number of times you want the observation duplicated). For example, the following command duplicates observation 5 one time:

```
dup 1 5
```

Alternatively, you can select the observation to copy by typing DUP on the command line, positioning the cursor on the desired observation, and pressing ENTER. By default, the observation is duplicated once. To duplicate the same observation again, leave the cursor on the command line and issue the DUP command again. Alternatively, you can follow the DUP command with the desired number of copies. For example, if the cursor is on the command line, the following command duplicates the most recently duplicated observation three times:

```
dup 3
```

If the cursor is positioned on an observation, then that observation is duplicated three times.

By default, the cursor is positioned on the leftmost displayed variable column of the first added observation. You can use the CURSOR command to select a particular variable to which the cursor is moved when the duplicate observation is displayed.

EDIT *data-set*<(*data-set-options*> <FORMULA=*SAS-catalog*<.*formula-entry*>>
opens another FSVIEW window and displays the specified data set for editing. The current FSVIEW window remains open, but the new FSVIEW window becomes the active window. All FSVIEW window commands are valid in the new window. When you use the END command to close the new window, you return to the FSVIEW window from which the EDIT command was issued.

Note: The EDIT command is not valid if the FSVIEW session is initiated by a PROC FSVIEW statement that includes the BROWSEONLY option. △

The additional FSVIEW window may overlay the current one. To control where the additional window appears, use the WREGION command before issuing the EDIT command. You can use the SWAP command to move among concurrent FSVIEW windows.

You can add data set options following the data set name. The options must be enclosed in parentheses. The FIRSTOBS= and OBS= options are ignored; all other data set options are valid. Refer to *SAS Language Reference: Dictionary* for a list and descriptions of data set options.

END
saves any changes that you make to the displayed data set (if it is open for editing), updates the FORMULA entry (if a formula name or formula catalog has been specified), closes the current FSVIEW window, and returns to the previous FSVIEW window.

If only one FSVIEW window is open, the END command ends the FSVIEW procedure.

FORMAT *variable-list 'format'* <*... variable-list-n 'format-n'*>
changes the format of one or more variables for display purposes only. The FORMAT command does not change any formats that are stored with the variables in the data set. Formats that are specified in the FORMAT command take precedence over formats that have been defined in the displayed data set or that are assigned in FORMAT statements.

The format name can be any SAS format, or it can be a custom format that you have defined with the FORMAT procedure. The format name must be enclosed in quotes.

You can specify multiple variables either individually or as a range. Separate two variable names with a dash to indicate a range. For example, the following command changes the display formats of variables X and Z only:

```
format x z 'date7.'
```

The following command changes the display format of all the variables between X and Z, inclusive:

```
format x-z 'date7.'
```

You can assign multiple formats in a single FORMAT command. For example, the following command assigns different display formats to the variables X and Z:

```
format x 'date7.' z 'monyy5.'
```

Note: Be aware that the format you assign a variable affects the informats you can assign with the INFORMAT statement. For example, suppose the data set that is displayed by the FSVIEW procedure contains a variable AMOUNT that is assigned the format DOLLAR10.2 but an informat of 10.2. Because of the format, values in the AMOUNT column are displayed with commas and a leading dollar sign, so the value 1250 would be displayed as $1,250.00. However, if you edit this value (for example, changing it to $1,150.00) and press ENTER, an error condition occurs. The 10.2 informat does not permit the dollar sign ($) or comma characters in entered values. An appropriate informat for this variable is COMMA., which does permit these characters. △

To remove a format assignment from the FORMULA entry so that the format that is defined in the data set is used, issue a FORMAT command that specifies the variable name but no format name. For example, to cancel the format for the variable X that is stored in the FORMULA entry, issue the following command in the FSVIEW window:

```
format x
```

FORMULA <*formula-name*>
reads in a FORMULA entry.

The general form of the *formula-name* argument is

<<*libref.*>*catalog-name.*>*entry-name*<.FORMULA>

You can specify a one-, two-, three-, or four-level name:

- If a one-level name is specified, it is treated as an entry name in the current formula catalog. The entry type is assumed to be FORMULA. If no catalog has been previously specified in a FORMULA command, in the FORMULA= option of the PROC FSVIEW statement, or in the *formula-name* argument of the FSVIEW command, then the procedure looks for the specified entry in

your personal PROFILE catalog (SASUSER.PROFILE, or WORK.PROFILE if the SASUSER library is not allocated.)

- If a two-level name is specified, the second level must be FORMULA. Any other value causes an error message. The name is treated as an entry name in the current formula catalog. If no formula catalog has been previously specified, the procedure looks for the specified entry in your personal PROFILE catalog.
- If a three-level name is specified, it is treated as *libref.catalog-name.entry-name*. The entry type is assumed to be FORMULA. You receive an error message if the specified libref is undefined or if the specified catalog does not exist.
- If a four-level name is specified, the fourth level must be FORMULA. Any other value causes an error message. You receive an error message if the specified libref is undefined or if the specified catalog does not exist.

If you omit the *formula-name* argument altogether, the procedure looks for an entry that has the default name. The default formula name is the name that was used in the previous FORMULA command. If no previous FORMULA command has been issued, then the default name is the name that was specified in the FORMULA= option of the PROC FSVIEW statement or in the *formula-name* argument of the FSVIEW command when the procedure was initiated. If no formula name was specified when the procedure was invoked, then the name of the displayed data set is used as the default FORMULA entry name. The procedure looks for the default entry in the current formula catalog. If no formula catalog has previously been specified, then the procedure looks for the default entry in your personal PROFILE catalog.

If the specified FORMULA entry exists, it replaces the FORMULA entry that is currently being used for the FSVIEW window. All current formula definitions are removed.

If the specified entry does not currently exist, then no entry is read in and a warning message is issued. An entry that has the specified name is created when you issue an END command to close the FSVIEW window, or when you issue a SAVE FORMULA command without specifying another formula name.

The FORMULA command is valid even if you do not use the FORMULA= option in the PROC FSVIEW statement or the *formula-name* argument in the FSVIEW command when you initiate the FSVIEW session.

FORWARD <*n* | HALF | PAGE | MAX>
scrolls vertically toward the bottom of the window. The following scroll amounts can be specified:

n	scrolls downward by the specified number of observations.
HALF	scrolls downward by half the number of lines in the window.
PAGE	scrolls downward by the number of lines in the window.
MAX	scrolls downward until the last observation is at the bottom of the window.

Note: See the discussion of the BOTTOM command for details about the restrictions that apply. △

If the scrolling increment is not explicitly specified, the window is scrolled by the amount that is specified in the VSCROLL parameter. The default VSCROLL amount is HALF.

Note: Regardless of the specified scroll increment, the FSVIEW procedure does not scroll beyond the last observation in the data set. △

HI <ON | OFF>
HI <*color* <*highlight*>>
HI <RESET <ALL>>
controls the highlighting status of an individual row or of the entire window, or sets the highlighting characteristics for an individual row or for the entire window.

To turn highlighting on for an unhighlighted row or to turn it off for a highlighted row, use the HI command. Type HI on the command line, position the cursor on the row that you want the command to affect, and press ENTER (or position the cursor on the desired row and press the function key to which the HI command has been assigned).

To turn highlighting on for all unhighlighted rows, use the HI ON command. To turn highlighting off for all highlighted rows, use the HI OFF command.

To set default highlighting characteristics for the entire window, follow the HI command with the desired color and (optionally) an attribute. The following values are valid for the *color* argument:

BLACK	CYAN	MAGENTA	RED
BLUE	GRAY	ORANGE	WHITE
BROWN	GREEN	PINK	YELLOW

The following values are valid for the *highlight* argument:

H (high intensity)	U (underlined)
R (reverse video)	B (blinking)

To change the highlighting characteristics for an individual row, type HI on the command line, followed by the new color and (optionally) the new attribute, position the cursor on the desired row, and then press ENTER.

To reset all rows to the default color and highlighting attribute, use the HI RESET ALL command.

To reset the highlight characteristics of an individual row to the default, type HI RESET on the command line, position the cursor on the desired row, and then press ENTER.

HSCROLL *n*
sets the default horizontal scrolling amount for the LEFT and RIGHT commands. For *n*, specify the number of variable columns to scroll by. The default amount is one column.

INFORMAT *variable-list 'informat'* <*... variable-list-n 'informat-n'*>
changes the informat of one or more variables in the FSVIEW window. The INFORMAT command does not change any informats that are stored with the variables in the data set. Informats that are specified with the INFORMAT command take precedence over informats that are defined in the data set or assigned in INFORMAT statements.

The informat name can be any SAS informat, or it can be a custom informat that you have defined with the FORMAT procedure. The informat name must be enclosed in quotes.

You can list multiple variable names individually or as a range. Separate two variable names with a dash to indicate a range. For example, the following command selects the informats for the variables X and Z only:

```
informat x z 'date7.'
```

The following command selects the informat for all the variables between X and Z, inclusive:

```
informat x-z 'date7.'
```

You can also assign multiple informats in a single INFORMAT command. For example, the following command selects different informats for the variables X and Z:

```
informat x 'date7.' z 'monyy5.'
```

Note: When using informats with the FSVIEW procedure, you should make sure the informats that you assign to variables are compatible with the formats for those variables. That is, the output that the format produces should be valid input for the informat. Otherwise, you complicate the process of editing the values of variables. For example, suppose the data set that is displayed by the FSVIEW procedure contains a variable AMOUNT that is assigned the informat 10.2 and the format DOLLAR10.2. Because of the format, values in the AMOUNT column are displayed with commas and a leading dollar sign, so the value 1250 would be displayed as $1,250.00. However, if you edit this value (for example, changing it to $1,150.00) and press ENTER, an error condition occurs. The 10.2 informat does not permit the dollar sign ($) or comma characters in entered values. An appropriate informat for this variable is COMMA., which does permit these characters. △

To remove an informat assignment from the FORMULA entry so that the informat that is defined the data set is used, issue an INFORMAT command that specifies the variable name but no informat name. For example, to cancel the informat for the variable X that is stored in the FORMULA entry, issue the following command in the FSVIEW window:

```
informat x
```

INITIAL <*obs* | CLEAR>
(editing command; not valid while browsing a data set)

selects an observation whose contents are used as initial values for added observations. By default, all variables in a new observation that is added by the autoadd feature are initialized with missing values. You can use this command to record the current variable values from an existing observation for use as initial values for new observations.

The values of the selected observation are recorded in the FORMULA entry. The recorded values are then used to initialize autoadded observations. Any subsequent changes that you make to the observation designated in the INITIAL command do not affect the stored values that are used to initialize new observations.

You can identify the desired observation by specifying its observation number as the *obs* argument in the INITIAL command, or you can type INITIAL on the command line, position the cursor on the desired observation, and press ENTER.

Use the INITIAL CLEAR command to reset the initial variable values for new observations to missing values.

KEYS
opens the KEYS window for browsing and editing function key definitions for the FSVIEW window. Function key definitions are stored in catalog entries of type KEYS.

The default function key definitions for the FSVIEW window are stored in the FSVIEW.KEYS entry in the SASHELP.FSP system catalog. If you are using this default set of key definitions when you issue the KEYS command and you change any key definitions in the KEYS window, a new copy of the FSVIEW.KEYS entry is created in your personal PROFILE catalog (SASUSER.PROFILE, or WORK.PROFILE if the SASUSER library is not allocated). The changes that you make are recorded in your personal copy of the KEYS entry.

LEFT <*n* | MAX>
scrolls the FSVIEW window to the left in increments of variable columns. The following scroll amounts can be specified:

n scrolls to the left by the specified number of variable columns.

MAX scrolls to the leftmost variable column.

If the LEFT command is used without arguments, the FSVIEW window is scrolled by the amount that is specified in the HSCROLL parameter. The default HSCROLL amount is one column.

MODIFY <RECORD | MEMBER>
UPDATE <RECORD | MEMBER>
changes the FSVIEW window from browsing to editing, or changes the control level of an FSVIEW window that is already open for editing.

Note: The MODIFY command is not permitted if the FSVIEW session is initiated with a PROC FSVIEW statement that includes the BROWSEONLY option. △

The data set can be opened for editing with either member- or record-level control. (Refer to "How the Control Level Affects Editing" on page 107 for more information on control levels.) You can specify a control level by using the RECORD or MEMBER arguments with the MODIFY command. If you do not use either argument, the default is record-level control (unless you use the CNTLLEV=MEMBER data set option with the data set name when you open the FSVIEW window).

When the FSVIEW window is open for editing, you can use the MODIFY RECORD and MODIFY MEMBER commands to change the current control level for the window.

The MODIFY command fails if the specified control level would cause a locking conflict. For example, you cannot specify MODIFY MEMBER if the same data set is open with record-level control in another FSVIEW window.

MOVE *variable* | *variable-range <target-variable>*
moves a variable column or a range of variable columns.

Note: The MOVE command affects only the order of variables in the FSVIEW window, not the actual position of the variables in the data set. △

To move a single variable, type MOVE on the command line, followed by the name of the variable to be moved; then place the cursor on the variable that you want the moved variable to follow and press ENTER.

To move a range of variables, specify the name of the first and last variables of the range, separated by a dash. All variables in the range must be of the same class, either ID or VAR. Indicate the variable that you want the range to follow by specifying the variable name as the *target-variable* argument. For example, the following command moves the variables VAR1 through VAR3 to the right of the variable VAR6:

```
move var1-var3 var6
```

You can also move a range of variables by typing MOVE and the variable range on the command line, positioning the cursor on the desired target variable, and pressing ENTER.

In order to move only one variable and specify the target variable as a command argument, you must supply the name of the moved variable as a range. For example, the following command moves the variable VAR2 after the variable VAR6:

```
move var2-var2 var6
```

NEW <(*data-set-options*)> <LIKE=*data-set*<(*data-set-options*)>> <FORMULA=*SAS-catalog*<.*formula-entry*>>
opens the FSVIEW NEW window for creating a new data set.

Note: An error message is returned if the data set that you specify in the NEW command already exists. You cannot use the NEW command to change variable names or attributes in an existing data set. △

In the FSVIEW NEW window you enter the names of your variables, the type, length, and (if desired) a label, format, and informat. See "Creating New SAS Data Sets" on page 128 for more information on using the FSVIEW NEW window. After you define the structure of the data set, an additional FSVIEW window is opened for entering values in the new data set.

You can add data set options following the data set name. The options must be enclosed in parentheses. Refer to *SAS Language Reference: Dictionary* for a list and descriptions of SAS data set options.

You can add the LIKE= option to specify an existing data set whose variable names and attributes are copied into the FSVIEW NEW window when it is opened. The data set name in the LIKE= option can also be followed by a list of data set options.

Note: If you use the LIKE= option to specify the data set that is currently displayed in the FSVIEW window, and if that data set is opened with member-level control, you receive an error message, and the characteristics of the displayed data set are not copied into the FSVIEW NEW window. The variable names and attributes of a data set cannot be read while the data set is opened with member-level locking. To avoid this problem, you can use the MODIFY RECORD command to change the displayed data set to record-level control before issuing a NEW command with a LIKE= option that specifies the displayed data set. △

You can add the FORMULA= option to specify a formula to be associated with the new FSVIEW window that is opened for the data set. The rules for the *formula-name* argument are the same as in the FORMULA= option of the PROC FSVIEW statement. See the description of the FORMULA option in "PROC FSVIEW Statement" on page 95 for details.

OBS
displays the number of the observation on which the cursor is currently positioned. This is useful when observation numbers are not displayed because an ID variable is used. Type OBS on the command line, position the cursor on an observation, and press ENTER. Or position the cursor on the desired observation and press the function key to which you assigned the OBS command.

PARMS
opens the FSVIEW Parameters window, from which general parameters for the FSVIEW session can be reviewed and modified. Refer to "Customizing the FSVIEW Environment" on page 136 for more information on the parameters that can be set in the FSVIEW Parameters window.

PROTECT ON | OFF <*variable* <*...variable-n*>>
(editing command; not valid while browsing a data set)
prevents changes from being made to the values in one or more variable columns when the data set is being edited.

You must follow the PROTECT command with either the ON or OFF argument, depending on whether you want to turn the protection feature on or off for the column. You can specify which column you want the command to affect by specifying the variable name as an argument for the command, or by typing PROTECT ON or PROTECT OFF on the command line, positioning the cursor on the desired column, and pressing ENTER.

Note: You cannot use the PROTECT OFF command to turn off protection for computed variables. △

You can control the protection feature for multiple columns with a single PROTECT command. You can provide a list of variable names, separated by spaces, as arguments for the command, or you can select a range of variable columns by specifying the names of the first and last variables in the range, separated by a dash. For example, the following command turns on protection for the variables X and Z:

```
protect on x z
```

The following command turns on protection for all variables between X and Z, inclusive:

```
protect on x-z
```

RENAME *current-variable-name new-variable-name*
changes the name for a specified variable column for the purposes of the FSVIEW window display only. The variable name in the data set is not changed; only the name that is used for the variable column in the FSVIEW window is changed. The new name is stored in a FORMULA entry rather than in the data set itself.

Like the original variable name, the new name is limited to 32 characters. All the other rules for SAS names are also applicable.

If the renamed variable is used in any formulas, the formulas are updated to reflect the new name.

RESET <ALL | *variable* <*... variable-n*>>
deletes some or all formula definitions. For computed variables, resetting the formula deletes the variable column from the FSVIEW window.

You can remove the formula definitions for more than one variable with a single RESET command. You can provide a list of variable names, separated by spaces, as arguments for the command, or you can select a range of variables by specifying the names of the first and last variables in the range, separated by a dash. For example, the following command resets formula definitions for the variables X and Z:

```
reset x z
```

The following command resets formula definitions for all variables between X and Z, inclusive:

```
reset x-z
```

Use the RESET ALL command to remove all current formula definitions.

REVIEW
opens the FSVIEW REVIEW window, in which all current formula definitions are displayed. See “Defining Formulas” on page 132 for more information on using the FSVIEW REVIEW window.

If no formulas are currently defined, the FSVIEW REVIEW window is not opened. Instead, a message in the FSVIEW window’s message line indicates that no formula definitions exist.

RIGHT <*n* | MAX>
scrolls the FSVIEW window to the right in increments of variable columns. The following scroll amounts can be specified:

n scrolls to the right by the specified number of variable columns.

MAX scrolls to the rightmost variable column.

If the RIGHT command is used without arguments, the FSVIEW window is scrolled by the amount that is specified in the HSCROLL parameter. The default HSCROLL amount is one column.

SAVE
stores all changes that have been made to the data set since the last time it was saved. See also the AUTOSAVE command.

SAVE FORMULA <*formula-name*>
stores all current formula definitions, general parameter settings, and FSVIEW window characteristics in a FORMULA entry.

The general form of the *formula-name* argument is

<<*libref.*>*catalog-name.*>*entry-name*<.FORMULA>

You can specify a one-, two-, three-, or four-level name:

- If a one-level name is specified, it is treated as an entry name in the current formula catalog. The entry type is assumed to be FORMULA. If no catalog has been previously specified in a FORMULA or SAVE FORMULA command, in the FORMULA= option of the PROC FSVIEW statement, or in the *formula-name* argument of the FSVIEW command, then the procedure stores the specified entry in your personal PROFILE catalog (SASUSER.PROFILE, or WORK.PROFILE if the SASUSER library is not allocated).
- If a two-level name is specified, the second level must be FORMULA. Any other value causes an error message. The name is treated as an entry name in the current formula catalog. If no formula catalog has been previously specified, the procedure stores the specified entry in your personal PROFILE catalog.
- If a three-level name is specified, it is treated as *libref.catalog-name.entry-name*. The entry type is assumed to be FORMULA. If the specified libref is undefined, you receive an error message. If the specified catalog does not exist, it is created.
- If a four-level name is specified, the fourth level must be FORMULA. Any other value causes an error message. You also receive an error message if the specified libref is undefined. If the specified catalog does not exist, it is created.

If you omit the *formula-name* argument altogether, the procedure stores the entry using the default name. The default formula name is the name that was used in the previous FORMULA command. If no FORMULA command has been issued, the default name is the name that was specified in the FORMULA= option

of the PROC FSVIEW statement or in the *formula-name* argument of the FSVIEW command when the procedure was initiated. If no formula name was specified when the procedure was invoked, then the name of the displayed data set is used as the default FORMULA entry name. The procedure stores the default entry in the current formula catalog. If no formula catalog has previously been specified, the procedure stores the default entry in your personal PROFILE catalog.

SETWSZ <CLEAR>
records the current size and position of the FSVIEW window in the FORMULA entry. The SETWSZ CLEAR command removes any previously recorded size and position information.

SHOW ID | VAR *variable* <...*variable-n*>
adds dropped variables to the FSVIEW window and controls how they are displayed. Dropped variables are variables that have been excluded from the FSVIEW window by using the DROP command, or variables that were omitted from the VAR and ID statements when the procedure was initiated.

You must follow the SHOW command with an argument that indicates whether the variables are to be displayed as ID variables or as scrolling variables. ID variables remain on the left side of the window as you scroll the other variable columns horizontally.

New scrolling variables are added to the right of existing scrolling variable columns. The first ID variable that you add replaces the observation number column. Additional ID variables are added to the right of existing ID variable columns, before the first scrolling variable column.

Note: The combined width for all ID variables cannot exceed 53 columns. If a single ID variable exceeds this length, it is truncated to 53 characters. If multiple ID variables are used, then only variables that can be displayed in 53 columns without truncation appear in the FSVIEW window. Any additional ID variables are not visible. △

You can add more than one variable with a single SHOW command. You can provide a list of variable names, separated by spaces, as arguments for the command, or you can specify a range of variable columns by specifying the names of the first and last variables in the list, separated by a dash. For example, the following command adds the variables X and Z as scrolling variables:

```
show var x z
```

The following command adds all variables between X and Z, inclusive, as scrolling variables:

```
show var x-z
```

In addition to adding variables that are currently dropped from the window, you can use the SHOW command to change current ID variables into scrolling variables and vice versa. For example, suppose that the FSVIEW window currently displays an ID variable that is named PRICE that you want to scroll with the other variables. Use the following command to redefine it as a scrolling variable:

```
show var price
```

Similarly, to change a scrolling variable into an ID variable, use the SHOW command to redefine it as an ID variable.

You can also issue the SHOW ID or SHOW VAR commands with no variables specified to open the Select Table Variables window. The Select Table Variables window provides a selection list from which the variables to be added can be chosen. See “Selecting Variables for FSVIEW Operations” on page 131 for more information on using the Select Table Variables window.

SMOOTH <ON | OFF>
turns smooth scrolling on and off, which determines how the FSVIEW window behaves when the scrollbar is used. When smooth scrolling is turned on, the contents of the FSVIEW window are refreshed as the thumb of the scrollbar is moved. When smooth scrolling is turned off, the contents of the FSVIEW window are not refreshed until the thumb of the scrollbar is released after it is moved.

When the SMOOTH command is used without the ON or OFF arguments, it acts as a toggle, turning the smooth scrolling feature on if it is currently off or off if it is currently on.

The current state of the smooth scrolling feature is recorded in the FORMULA entry when the entry is saved. The saved state takes effect when the FORMULA entry is loaded.

SORT <ASCENDING | DESCENDING> *variable* <...<ASCENDING | DESCENDING> *variable-n*>
(editing command; not valid while browsing a data set)

sorts the data set by the specified variables and saves the data set. If you have deleted any observations from the data set, the remaining observations are renumbered sequentially when you issue the SORT command.

CAUTION:
The SORT command sorts the data set itself, not just the displayed values. The FSVIEW procedure sorts a data set in place. It does not leave an unsorted copy of the original. If you need to preserve a copy of the original data, make a copy of the data set before using the SORT command.

If the KEEP= or DROP= data set options were specified with the PROC FSVIEW statement or FSVIEW command, then only the specified variables will be retained in the data set after it is sorted. △

Note: The SORT command is not valid while a permanent or temporary WHERE clause is in effect. △

By default, the procedure sorts the data set in ascending order of each specified variable. To sort a data set in descending order of a particular variable, precede the variable name with the DESCENDING option. You must specify the DESCENDING option separately for each variable. For example, the following command sorts the data in descending order by STATE and in descending order by CITY within STATE:

```
sort descending state descending city
```

You can also use the ASCENDING option to explicitly indicate the default sorting order.

If the sort fails for any reason, the FSVIEW procedure is immediately terminated in order to preserve the contents of the data set.

If your site or SAS session uses a host sort utility rather than the sort utility that is supplied by the SAS System, you may not be able to use your system's attention key to cancel a sort that is in progress.

TOP
scrolls the FSVIEW window vertically so that the first observation in the data set appears at the top of the window.

You can interrupt the TOP scroll operation before the first observation is reached. This feature is useful when you want to halt a scroll request while browsing or editing a large data set. To halt a TOP scroll operation that is in progress, press the interrupt key or key combination for your system. The FSVIEW procedure displays a requestor window that gives you the options of canceling or resuming the scrolling operation.

Note: The key or key combination that is used to interrupt an active process depends on your host operating system and terminal device. For example, some systems have a key labeled BREAK or ATTENTION (or ATTN). Other systems use a combination of the CTRL key and another key. Refer to your host documentation to determine the interrupt key or key combination for your host operating system and terminal device. △

UPDATE <RECORD | MEMBER>
See the MODIFY command.

VSCROLL *n* | HALF | PAGE
sets the default vertical scrolling amount for the FORWARD and BACKWARD commands. The following scroll amounts can be specified:

n scroll by the specified number of lines.

HALF scroll by half the number of lines in the window.

PAGE scroll by the full number of lines in the window.

The default amount is HALF.

WHERE <<ALSO> *expression*> | <UNDO | CLEAR>
imposes one or more sets of conditions that observations in the data set must meet in order to be displayed. *Expression* is any valid WHERE expression that includes one or more of the variables in the input data set. Refer to the description of the WHERE statement in *SAS Language Reference: Dictionary* for details about the operators and operands that are valid in WHERE expressions. Observations that do not satisfy the specified conditions cannot be displayed or edited.

The complete set of conditions is called a temporary WHERE clause. The conditions can be modified or canceled during the FSVIEW session. In contrast, the WHERE statement defines a permanent WHERE clause that cannot be changed or canceled during the FSVIEW session and which is not affected by WHERE commands. See “WHERE Statement” on page 101 for details.

The word *Where...* appears in the upper-right corner of the window whenever a temporary WHERE clause is in effect.

The WHERE command has several forms:

WHERE *expression*
applies the conditions that are specified in *expression* as the new temporary WHERE clause, replacing any clause that was previously in effect.

WHERE ALSO *expression*
adds the conditions that are specified in *expression* to the existing temporary WHERE clause.

WHERE UNDO
deletes the most recently added set of conditions from the temporary WHERE clause.

WHERE
WHERE CLEAR
cancels the current temporary WHERE clause.

If you edit values in an observation so that it no longer meets the conditions of the WHERE clause, that observation can still be displayed and edited. However, a warning message is printed whenever the observation is displayed, indicating that the observation no longer meets the WHERE conditions.

If you use the ADD or DUP command to add a new observation and then enter values that do not meet the WHERE conditions, that observation cannot be displayed or edited once you scroll to another observation.

Creating New SAS Data Sets

You can use the FSVIEW procedure to create new SAS data sets. You name the variables and specify their attributes in the FSVIEW NEW window. After you exit the FSVIEW NEW window, the data set is created. It contains one observation, which is filled with missing values. An FSVIEW window is then automatically opened for editing so that you can enter values in the new data set.

Opening the FSVIEW NEW Window

You can open the FSVIEW NEW window in two ways:

- by using the NEW= option in the PROC FSVIEW statement when you invoke the procedure
- by using the NEW command after you enter the procedure.

Using the NEW= Option

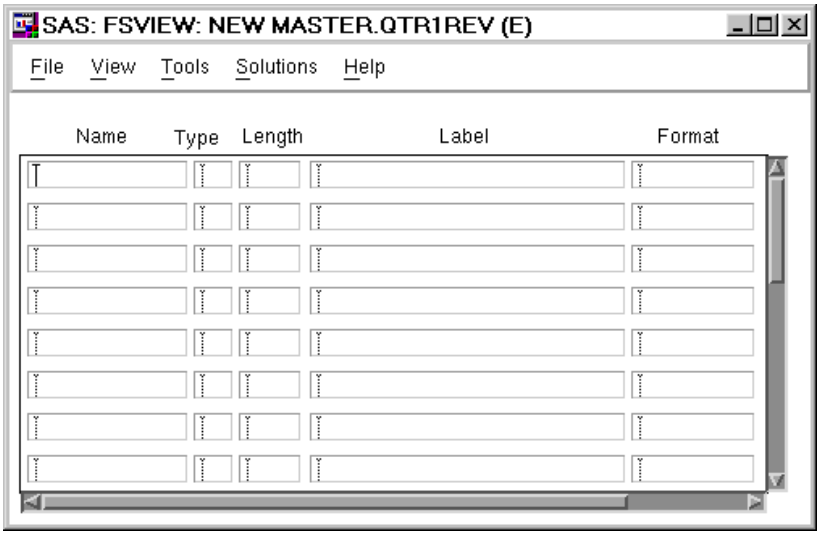
When you invoke the FSVIEW procedure with the NEW= option rather than with the DATA= option in the PROC FSVIEW statement, the FSVIEW NEW window is opened when the FSVIEW procedure is initiated.

For example, the following statements initiate an FSVIEW session and open the FSVIEW NEW window to create the new data set MASTER.QTR1REV:

```
proc fsview new=master.qtr1rev;
run;
```

Display 8.2 on page 128 shows the initial FSVIEW NEW window display.

Display 8.2 The FSVIEW NEW Window



You define the variables in the fields provided. When you issue the END command, the data set is created, and the FSVIEW window is automatically opened for editing so that you can add observations and enter variable values.

Note: After you issue the END command to close the FSVIEW NEW window, you cannot return to that window to make changes to information in the data set. △

Using the NEW Command

You can issue the NEW command from an FSVIEW window to define a new data set within an FSVIEW session.

For example, the following command opens the FSVIEW NEW window to define the data set MASTER.QTR1REV:

```
new master.qtr1rev
```

The display in this case is the same as shown in Display 21.2. However, when you use the END command to close the FSVIEW NEW window and create the new data set, an additional FSVIEW window is opened. You can use that window to add observations and enter values for the new data set. The new FSVIEW window may overlay the window from which the NEW command was issued.

Creating a Data Set That Is Like an Existing One

If you want to create a new SAS data set that is identical or similar in structure to an existing data set, you can save time by letting the SAS System do some of the work. Instead of entering all the variable information, use the LIKE= option to identify an existing SAS data set. When the FSVIEW NEW window is opened, the variable names and attributes of the data set that you specified in the LIKE= option are automatically displayed.

The display contains all the information necessary to create a SAS data set that has exactly the same structure as the specified data set. (Only the structure of the specified data set is copied, not the contents.) You have the option of making changes to the variable names and attributes before creating the new data set. You can also delete a variable entirely by blanking out its name.

You open the FSVIEW NEW window in one of two ways: either with the NEW= option when you invoke the FSVIEW procedure or with the NEW command when you open an additional FSVIEW window. You can use the LIKE= option with both of these methods.

In the PROC FSVIEW statement, the LIKE= option must always be used in conjunction with the NEW= option. For example, the following statements initiate an FSVIEW session, open the FSVIEW NEW window to create the data set MASTER.QTR2REV, and fill in the fields of the FSVIEW NEW window with the names and attributes of the variables in the data set MASTER.QTR1REV:

```
proc fsview new=master.qtr2rev
     like=master.qtr1rev;
run;
```

The following command performs the same tasks, but from within an active FSVIEW session:

```
new master.qtr2rev like=master.qtr1rev
```

Closing the FSVIEW NEW Window

To close the FSVIEW NEW window, create the SAS data set, and open the FSVIEW window so that you can enter values for each observation, issue the END command. To close the FSVIEW NEW window without creating a data set, issue the CANCEL command.

FSVIEW NEW Window Commands

In addition to the global commands that are discussed in Chapter 9, "SAS/FSP Software Global Commands," on page 143, you can use the following commands in the FSVIEW NEW window to scroll information, to duplicate selected lines, or to exit with the choice of creating a data set or canceling it.

Scrolling

BACKWARD <HALF | PAGE | *n*>

BOTTOM

FORWARD <HALF | PAGE | *n*>

LEFT

RIGHT

TOP

Exiting

CANCEL

END

Command Descriptions

Here are descriptions of the FSVIEW NEW window commands:

BACKWARD <*n* | HALF | PAGE>
scrolls vertically toward the top of the window. The following scroll amounts can be specified:

n	scrolls upward by the specified number of lines.
HALF	scrolls upward by half the number of lines in the window.
PAGE	scrolls upward by the number of lines in the window.

If you do not explicitly specify an amount to scroll, the default increment is HALF.

BOTTOM
scrolls downward until the last line that contains a variable definition is displayed.

CANCEL
closes the FSVIEW NEW window and ends the FSVIEW session. The new data set is not created.

END
closes the FSVIEW NEW window, creates the SAS data set that was defined in the window, and opens an FSVIEW window for editing observations in the newly created data set.

FORWARD <*n* | HALF | PAGE>
scrolls vertically toward the bottom of the window. You can scroll forward only if you have filled the last blank variable-definition line that is currently displayed or if there are more variables to be displayed. The following scroll amounts can be specified:

n	scrolls downward by the specified number of lines.

HALF scrolls downward by half the number of lines in the window.

PAGE scrolls downward by the number of lines in the window.

If you do not explicitly specify an amount to scroll, the default increment is HALF.

LEFT
displays the FORMAT column when the INFORMAT column is displayed or vice versa.

RIGHT
displays the INFORMAT column when the FORMAT column is displayed or vice versa.

TOP
scrolls upward until the first variable-definition line is displayed.

Duplicating Existing SAS Data Sets

The FSVIEW procedure enables you to create a new data set that duplicates both the structure and the contents of the data set that is currently displayed in the FSVIEW window. You use the CREATE command to create new data sets.

The new data set can incorporate some or all of the variables from the currently displayed data set. (Dropped variables cannot be used.) Computed variables in the displayed data set become data set variables in the new data set. The current computed values of the variables are stored as the values of the new variables. If a temporary or permanent WHERE clause is in effect, then only observations from the displayed data set that satisfy the WHERE conditions are copied to the new data set.

There are two ways to specify which variables in the displayed data set are used in the new data set:

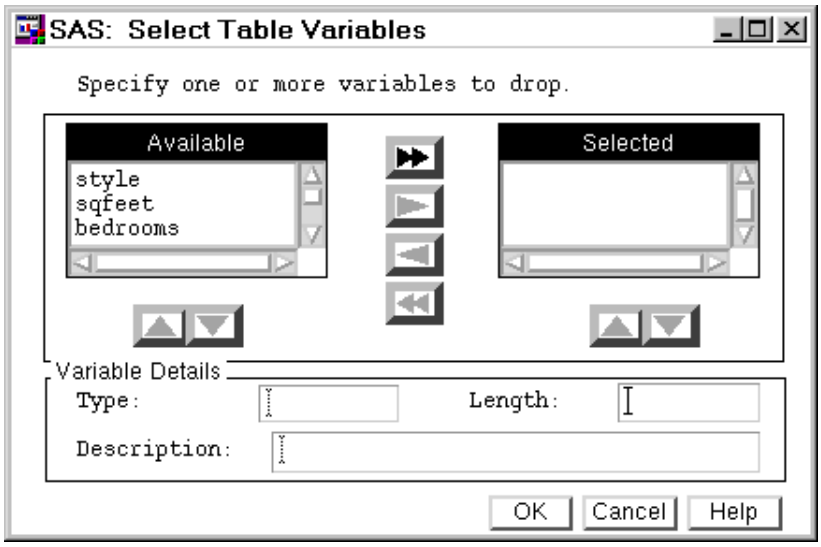
- You can list the desired variable names as arguments in the CREATE command (or use the ALL argument to copy all variables).
- You can issue a CREATE command with no variable names listed or with the **?** argument. This form of the command opens the Select Table Variables window, from which you can select the variables to include in the new data set. By default, all variables in the displayed data set are selected in the Select Table Variables window. The Select Table Variables window is explained in the following section.

Selecting Variables for FSVIEW Operations

The CREATE, DROP, and SHOW commands in the FSVIEW window can open the Select Table Variables window so that you can select which variables from the displayed data set are included in the specified operation. The Select Table Variables window is a convenient alternative to entering variable names as command arguments.

Display 8.3 on page 132 shows a typical Select Table Variables window.

Display 8.3 The Select Table Variables Window



The Select Table Variables window includes lists of the available and selected variables, along with control objects for moving either individual variables or all variables between the lists. Use the single-arrow controls to move individual variables, and use the double-arrow controls to move all variables.

Below each list are control objects that you can use to change the relative position of variables within the list (for commands like SHOW where the order of variables in the list is significant). To change the order of the variables within one of the lists, select the variable you want to move. Then select the corresponding up-arrow control to move the variable toward the top of the list, or select the down-arrow control to move it toward the bottom. As an added feature, the type, size, and label of the selected variable are displayed in the Variable Details section whenever you select a variable in either list.

After you have moved all the desired variables to the **`Selected`** list, select OK to complete the operation. You can select Cancel to cancel the operation.

Defining Formulas

The FSVIEW procedure enables you to define formulas that perform computations and otherwise manipulate variables. *Formulas* are SAS expressions that calculate values for variables in the FSVIEW window. You can define formulas for variables in the data set, or you can create computed variables to display the formula results.

The effect of assigning a formula to a data set variable depends on whether the FSVIEW window is open for browsing or editing. If the window is open for browsing, then the result of computations performed by the assigned formula is displayed in the variable's column in the FSVIEW window, but the variable value in the data set is not affected. If the window is open for editing, then the result of the formula computations replaces the value that is stored in the data set as well as the value in the FSVIEW window.

Computed variables exist only in the FSVIEW window to show the result of computations that are performed by a formula; they are not added to the displayed data set. The values of computed variables cannot be edited. However, within the FSVIEW window computed variables can be manipulated just like data set variables. The only exception is that you cannot use the SORT command to sort the data set according to the values of a computed variable.

Use the DEFINE command to define formulas. The FSVIEW procedure provides two ways to use the DEFINE command:

- Use the DEFINE command alone or with only the *variable-name* argument to open the FSVIEW Define window. In this window you can enter all the information necessary to define the formula. You can also use this form of the command to open the FSVIEW Define window to edit an existing formula. See "Defining Formulas in the FSVIEW Define Window" on page 134 for more information on using the FSVIEW Define window.
- Supply the formula as an argument with the DEFINE command. Separate the formula from the variable name with an equal sign (=).

Formulas are stored in SAS catalog entries that have the type FORMULA. See "Creating FSVIEW Applications" on page 140 for details.

How Formulas Are Evaluated

Formulas are evaluated in the order in which the corresponding variables appear in the FSVIEW window. It is the order of the variables, not the order in which the formulas are defined, that determines the evaluation order. Formulas are evaluated first for any ID variables, then for the remaining variables in left-to-right order. After the formulas for displayed variables are evaluated, any formulas assigned to variables that are dropped from the FSVIEW window are evaluated, in the order in which the variables are dropped.

If you use a computed variable in the formula definition for another variable, the variable that is named in the formula must appear in the FSVIEW window to the left of the variable for which the formula is defined. Otherwise, the value of the variable in the formula is missing when the formula is evaluated.

When the FSVIEW window is displayed or redrawn, the formulas are evaluated for all displayed observations. When an observation is edited, formulas are evaluated for the edited observation only.

Using SAS Component Language in Formulas

Formulas can also include any valid SAS Component Language functions or statements except those that are valid only in SAS/AF software and the following statements:

CONTROL ALWAYS	ERROROFF	PROTECT
CURSOR	ERRORON	UNPROTECT

The MODIFIED, ERROR, and CUROBS functions are valid.

Each formula that you define is treated as a block of SCL code. When you define a formula, the FSVIEW procedure internally adds a label for the formula and a RETURN statement so that each formula can be treated as a separate SCL block. (The label is the same as the variable name.) The SCL code segments are compiled individually when the formulas are defined.

For example, suppose you define a computed variable that is named MODSTAT with the following formula:

```
modstat=modified(begdate)
```

The SCL block for the MODSTAT variable would be stored as follows:

```
MODSTAT: modstat=modified(begdate);RETURN;
```

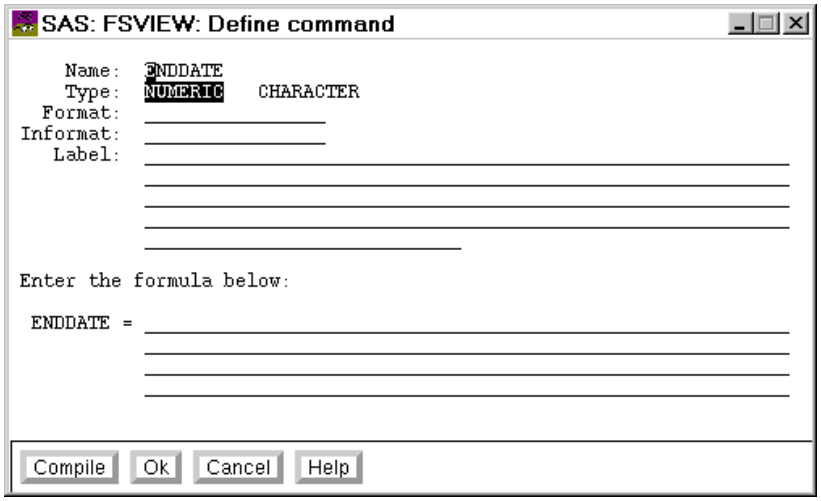
The blocks are executed in the order in which the corresponding variables appear in the FSVIEW window.

Refer to *SAS Component Language: Reference* for details about including SAS Component Language in FSVIEW formulas.

Defining Formulas in the FSVIEW Define Window

When you issue a DEFINE command in the FSVIEW window without specifying a formula for the variable, the FSVIEW Define window is opened so that you can enter the formula definition and other variable attributes. Display 8.4 on page 134 shows the FSVIEW Define window that is opened for a variable named ENDDATE:

Display 8.4 The FSVIEW Define Window



The FSVIEW Define window includes the following fields:

Name
is the name of the variable with which this formula is associated. You can change the name in this field while the FSVIEW Define window is open to change the variable for which the formula is defined. If you specify the name of a variable that has an existing formula definition, the remaining fields in this window are filled with values from the current definition.

Type
is the variable type. To select a type, position the cursor on the corresponding term and press ENTER. The default is NUMERIC.

Format
is the format that determines how the variable is displayed. The format also determines the width of the variable's column in the FSVIEW window. If you do not specify a format, a default is used:

- BEST12. for numeric variables
- $8. for character variables.

Informat

is the informat that determines how values that are entered in the variable's column in the FSVIEW window are interpreted. If you do not specify an informat, a default is used:

- 12. for numeric variables
- $8. for character variables.

Label
is a descriptive label up to 256 characters long. (The FSVIEW procedure does not display variable labels.)

At the bottom of the window are four lines for entering the formula. The FSVIEW Define window supplies the variable name and the equal sign that begin the definition.

Formulas that you define in the FSVIEW Define window can consist of more than one statement. If you use multiple statements, separate the statements with semicolons. The definition can include as many statements as you can fit in the space provided.

The four lines of text in the FSVIEW Define window are seen as a single line of text when the formula is compiled. Spaces are not automatically inserted between the end of one line in the window and the beginning of the next. If you do not leave spaces between words at the line breaks, you may encounter syntax errors when the formula is compiled.

The following section describes the commands you can use in the FSVIEW Define window.

FSVIEW Define Window Commands

In addition to the global commands that are discussed in Chapter 9, "SAS/FSP Software Global Commands," on page 143, you can use the following commands in the FSVIEW Define window:

CANCEL
closes the FSVIEW Define window without recording the formula or variable attributes that were entered in the window. Once you open the FSVIEW Define window, you must use this command if you want to close the window without entering a formula definition.

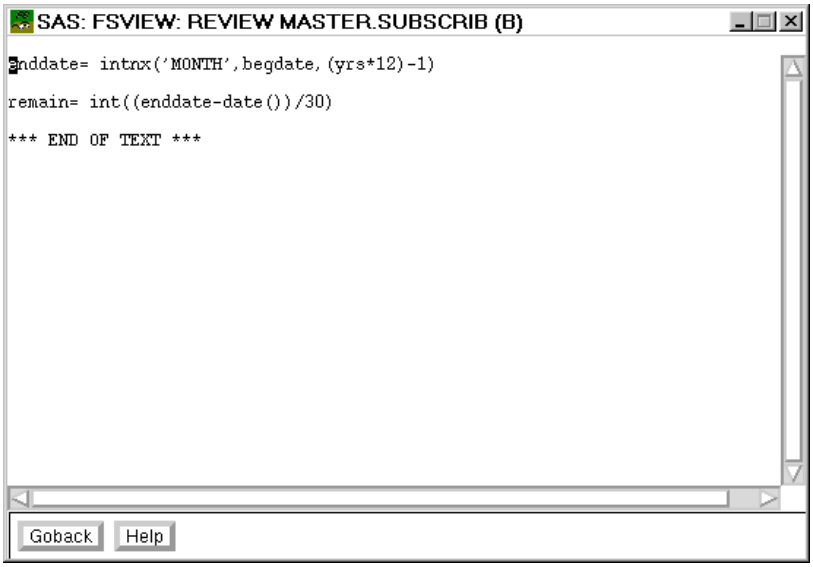
COMPILE
compiles the current formula and reports whether it is a valid SAS expression.

END
compiles the current formula. If the formula is valid, the FSVIEW Define window is closed, the formula definition is recorded in the FORMULA entry, and you return to the FSVIEW window. If the formula contains errors, an error message is displayed and the FSVIEW Define window remains open.

Reviewing Formula Definitions

You can browse existing formula definitions in the FSVIEW REVIEW window. Use the REVIEW command in the FSVIEW window to open the FSVIEW REVIEW window. Display 8.5 on page 136 shows a FSVIEW REVIEW window that contains two formula definitions.

Display 8.5 Viewing Formulas in the FSVIEW REVIEW Window



The FSVIEW REVIEW window displays the formulas for browsing only. You cannot make changes to the formulas in this window. The formula definitions are listed in the order in which they are entered.

FSVIEW REVIEW Window Commands

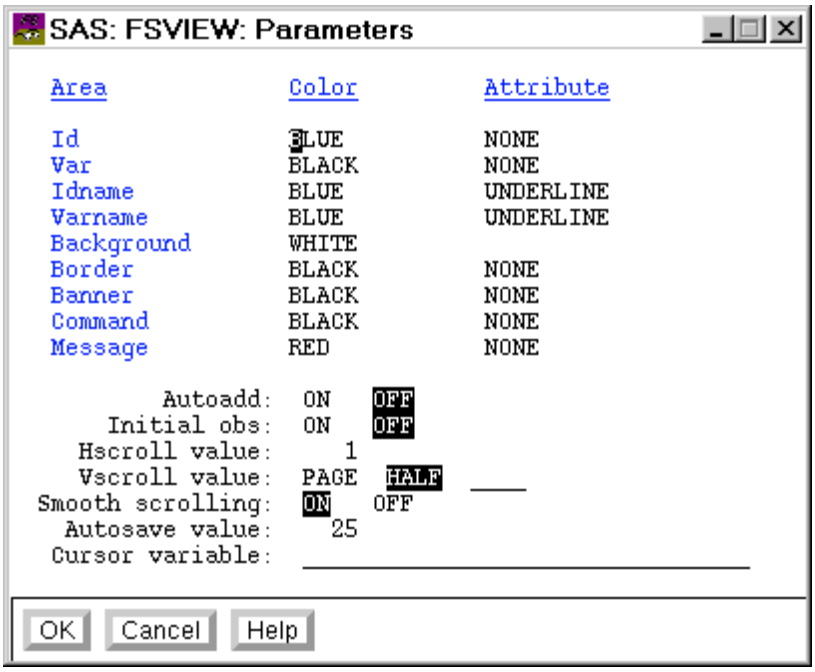
The FSVIEW REVIEW window uses the SAS text editor. All text editor browsing commands are valid in this window. Refer to the online Help for Base SAS software for more information on the commands that can be used in text editor windows.

Use the END command to close the FSVIEW REVIEW window and return to the FSVIEW window.

Customizing the FSVIEW Environment

You can control some of the characteristics of the FSVIEW session by specifying values for parameters in the FSVIEW Parameters window. Use the PARMS command to open the FSVIEW Parameters window. Display 8.6 on page 137 shows the FSVIEW Parameters window with the default parameter settings.

Display 8.6 The FSVIEW Parameters Window



You can change the current parameter settings in the FSVIEW Parameters window. Some of the parameter settings in the FSVIEW Parameters window are choice groups. To turn on these parameters, move the cursor to the term for the desired parameter status and press ENTER. The selected term is highlighted. Other parameter settings are fields in which you enter the desired parameter values. The following section describes the parameters that are available in the FSVIEW Parameters window.

If you create a FORMULA entry for the FSVIEW session, all the parameter settings from the FSVIEW Parameters window are recorded and will be in effect the next time that FORMULA entry is loaded.

Parameter Descriptions

The following parameters can be controlled by using the FSVIEW Parameters windows:

Color and Attribute
: specify the color and highlighting attributes of the following areas of the FSVIEW window:

ID
: controls the color of observation numbers or variable values in the ID columns.

Var
: controls the color of variable values in the variable columns.

IDname
: controls the color of the column headings for the ID columns.

Varname
: controls the color of the column headings for the variable columns.

Background
: controls the background color of the FSVIEW window.

Border
: controls the color of the window border in character-based display environments.

Note: This parameter has no effect in graphical windowing environments. △

Banner
: controls the color of the **Command===>** text at the left of the command line.

  Note: This parameter has no effect if a menu bar is displayed in place of a command line. △

Command
: controls the color of the text that you type on the command line.

  Note: This parameter has no effect if a menu bar is displayed in place of a command line. △

Message
: controls the color of text that is displayed in the window's message line.

The following values are valid in the **Color** fields:

BLUE	GREEN	WHITE	ORANGE
RED	CYAN	GRAY	
PINK	YELLOW	BROWN	

If a specified color is not available on a user's device, the procedure substitutes the available color that most closely matches the specified color. Some devices do not support changing the background color; for these devices, the background color parameter is ignored.

The following values are valid in the **Attribute** fields. (You can abbreviate the values by entering only the first letter; the procedure fills in the complete value when you press ENTER.)

HIGHLIGHT	high intensity
BLINKING	blinking
UNDERLINE	underlined
REVERSE	reverse video

If a parameter specifies a highlighting attribute that is not available on the device, the parameter is ignored. You cannot specify a highlighting attribute for the background.

If you use the COLOR command in the FSVIEW window, the colors and highlighting attributes you specify using that command appear in these parameters.

Autoadd
: controls whether new observations are automatically added to the displayed data set. This parameter reflects the current status of the autoadd feature, which can also be controlled with the AUTOADD command in the FSVIEW window.

Initial obs
: controls whether initial values have been recorded to fill added observations. This parameter is set on when you use the INITIAL command in the FSVIEW window.

The INITIAL command records values from an existing observation so that they can be copied into new observations that are added by the autoadd feature.

Hscroll value
controls how many columns are scrolled when a LEFT or RIGHT command does not specify an explicit scroll amount. The default value is 1.

If you use the HSCROLL command in the FSVIEW window, the value that you specify with that command appears in the field for this parameter.

Vscroll value
controls how many observations are scrolled when a FORWARD or BACKWARD command does not specify an explicit scroll amount. The following default scroll amounts can be specified:

n	scroll by the specified number of lines. Enter the desired value in the space provided; then press ENTER to select this value. The value is highlighted in reverse video when it is selected.
HALF	scroll by half the number of lines in the window.
PAGE	scroll by the number of lines in the window.

The default value is HALF.

If you use the VSCROLL command in the FSVIEW window, the value that you specify with that command is shown in the field for this parameter.

Smooth scrolling
controls whether smooth scrolling is available. When smooth scrolling is turned on, the contents of the FSVIEW window are refreshed as the thumb of the scrollbar is moved. When smooth scrolling is turned off, the contents of the FSVIEW window are not refreshed until the thumb is released. If you use the SMOOTH command in the FSVIEW window, the scrolling status that you specify appears in this parameter.

Autosave value
controls how frequently the data set is automatically saved. An automatic save is performed whenever the number of modifications (additions, deletions, or changes) specified in this parameter have been made since the last save. The default value is 25.

If you use the AUTOSAVE command in the FSVIEW window, the value that you specify with that command appears in the field for this parameter.

Cursor variable
controls which variable column the cursor is positioned on when a new observation is added by the autoadd feature or by the DUP command.

If you use the CURSOR command in the FSVIEW window, the variable that you specify with that command appears in the field for this parameter.

FSVIEW Parameters Window Commands

In addition to the global commands that are discussed in Chapter 9, "SAS/FSP Software Global Commands," on page 143, you can use the following commands in the FSVIEW Parameters window:

CANCEL
closes the FSVIEW Parameters window without recording any changes to the previous parameter settings.

END
closes the FSVIEW Parameters window and applies the parameter settings that it contains to the FSVIEW window.

Creating FSVIEW Applications

If you are an applications developer, you can use the FSVIEW procedure as the basis for data entry applications and editing applications. The FSVIEW procedure enables you to customize the application environment to suit the needs of your users. Customization can include

- redesigning the display
- defining formulas for creating computed variables or for manipulating data set variables
- setting general parameters that control the behavior of the FSVIEW session.

The feature of the FSVIEW procedure that makes this customization possible is the FORMULA entry. *Formula entries* are SAS catalog entries of type FORMULA that are created by the FSVIEW procedure to record the following information about the FSVIEW session:

- the names and order of variables in the FSVIEW window.
- any formula definitions for the variables.
- the format and informat of variables for which specific formats or informats are assigned, using the FORMAT or INFORMAT commands in the FSVIEW window.

  Note: Formats and informats that are specified in the data set or by using FORMAT and INFORMAT statements in conjunction with the PROC FSVIEW statement at procedure invocation are not recorded in the FORMULA entry. △

  The format and informat for computed variables are always stored in the FORMULA entry. If a FORMAT or INFORMAT command for a computed variable does not specify a valid format or informat, it is ignored.
- the current size and position of the FSVIEW window. (These values can be reset with the SETWSZ command.)
- the current FSVIEW window colors.
- the initial values for variables in added observations. (These values can be reset with the INITIAL command.)
- the current settings of all general parameters: Autoadd, Autosave value, Hscroll value, Vscroll value, and Cursor variable.

Creating and Updating Formula Entries

If you specify a catalog name or a complete formula name when you open an FSVIEW window (by using the FORMULA= option in the PROC FSVIEW statement, the *formula-name* argument in the FSVIEW command, or the FORMULA= option in the BROWSE, EDIT, and NEW commands in the FSVIEW window), a FORMULA entry is created automatically when the FSVIEW window is closed. You can also create a FORMULA entry by issuing a SAVE FORMULA command in the FSVIEW window. If you specify only a catalog name, the FSVIEW procedure uses the name of the displayed data set as the formula name.

Loading Formula Entries

To load an existing FORMULA entry when you invoke the FSVIEW procedure, use the FORMULA= option in the PROC FSVIEW statement or the *formula-name* argument in the FSVIEW command. You can also use the FORMULA= option in the BROWSE, EDIT, and NEW commands in the FSVIEW window to load a FORMULA entry for additional FSVIEW windows that are opened during the FSVIEW session. You can use the FORMULA command in an open FSVIEW window to change the FORMULA entry that is used for the window or to load a FORMULA entry if none was used when the window was originally opened. (The FORMULA command is valid even if you do not specify a formula catalog when you invoke the FSVIEW procedure.)

CHAPTER

9

SAS/FSP Software Global Commands

Overview

The following commands are available in all SAS/FSP windows. Details about these commands are provided in this chapter.

Procedure Environment	DEBUG	SETWDATA
	MSG	SETWNAME
	SETCR	SHOWTYPE
	SETHELP	TYPE
	SETPMENU	WSIZE
Concurrent Windows	SWAP	WREGION
Printing	FONT	PRTFILE
	FORMNAME	SPRINT

The following SAS windowing environment global commands are also valid in all SAS/FSP windows. SAS global commands are discussed in the online Help for Base SAS software.

Window Call	AF|AFA	HELP
	ASSIST	LIBNAME
	CATALOG	LOG
	DIR	MANAGER|MGR
	EFI	MENU
	EIS	NOTEPAD
	FILENAME	OPTIONS
	FOOTNOTES	OUTPUT|OUT
	FSBROWSE	PROGRAM|PGM
	FSEDIT	SASDOC
	FSFORM	TITLES
	FSLETTER	VAR
	FSLIST	VIEWTABLE
	FSVIEW	
Window Management	AUTOPOP	PMENU
	BYE|ENDSAS	PREVCMD|?
	COLOR	PREVWIND
	COMMAND	RESHOW
	HOME	SCROLLBAR
	ICON	STATUS
	KEYDEF	X
	NEXT	ZOOM
Window Size and Position	CASCADE	TILE
	RESIZE	WDEF
Cut and Paste Facility	CUT	PLIST
	MARK	SMARK
	PASTE	STORE
	PCLEAR	UNMARK

Procedure Environment Commands

DEBUG <ON | OFF>
turns the SAS Component Language source-level debugger on or off. If you use the DEBUG command without an argument, the debugger is turned on. You cannot turn the debugger off while a debugger command is active.

The SCL debugger is a tool for identifying and correcting problems in SAS Component Language programs and FSVIEW formulas. Refer to *SAS Component Language: Reference* for information on using the SCL debugger.

MSG

opens the LOG window or makes the LOG window the active window if it is already open. Error and warning messages that are generated by the procedures in SAS/FSP software are written to the LOG window.

SETCR <STAY | HTAB | VTAB | NEWL | CMPDN RET | NORET MOD | NOMOD>

defines the behavior of the ENTER key. You can use the command without arguments to report the current settings. If you specify arguments, you must supply one from each of the following three groups:

Cursor Movement

determines the behavior of the cursor when the ENTER key is pressed.

STAY	The cursor does not move.
HTAB	The cursor moves to the next unprotected field on the current line. If there are no more unprotected fields on the current line, the cursor moves to the first unprotected field on a line below the current field.
VTAB	The cursor moves to the next unprotected field below the current field.
NEWL	The cursor moves to the first unprotected field on a line below the current line.
CMPDN	The cursor moves to the command line.

Control Passing

determines whether control passes to the application when the ENTER key is pressed.

RET	Control returns to the application.
NORET	Control does not return to the application.

Field Modification

determines whether a field is marked as modified when the ENTER key is pressed while the cursor is on the field, even if the field value has not been changed.

MOD	The field is marked as modified.
NOMOD	The field is not marked as modified unless the field value is changed.

SETHELP <*libref.catalog-name.*>*entry-name*<*.entry.type*>

identifies the source of Help information for the current window. When you issue the HELP command in a window, the SAS System opens a HELP window and displays information from a catalog entry of type HELP or CBT. The SETHELP command assigns a particular HELP or CBT entry to the current window.

If you omit the *libref* and *catalog-name* arguments, the procedure looks for the specified entry first in the current catalog (if a catalog is currently open), then in your personal PROFILE catalog (SASUSER.PROFILE or WORK.PROFILE), and finally in the system catalog SASHELP.FSP. The *entry-type* value can be either CBT or HELP. If you omit the type, the procedure looks for an entry of type CBT.

You can use the BUILD procedure in SAS/AF software to create custom Help entries for SAS/FSP windows. You can create either HELP or CBT entries, depending on the level of interactivity you want to provide. HELP entries provide only text. CBT entries can provide more sophisticated assistance, including topic selection lists and nested windows. Refer to *SAS/AF Procedure Guide* for more information on creating HELP and CBT entries.

SETPMENU <*libref.catalog-name*.>*menu-name*
identifies the source of menu definitions for the current window. Characteristics of menu bars and pull-down menus are stored in catalog entries of type PMENU. The SETPMENU command associates a particular PMENU entry with the current window.

If you omit the *libref* and *catalog-name* arguments, the procedure looks for the PMENU entry first in the current catalog (if a catalog is currently open), then in your personal PROFILE catalog (SASUSER.PROFILE or WORK.PROFILE), and finally in the system catalog SASHELP.FSP.

You can use the PMENU procedure in Base SAS software to create custom menu bars and pull-down menus for SAS/FSP windows. Refer to *Base SAS Procedures Guide* for more information about the PMENU procedure.

The SETPMENU name only associates a PMENU entry with the current window. It does not turn on menus if they are not currently active. Use the PMENU ON command to turn on menus for all SAS System windows, or use the COMMAND command to replace the command line of the current window with a menu bar.

SETWDATA *data-text*
specifies additional text to display on the window's title bar following the window title and separated from the title by a dash (-).

SETWNAME *title-text*
specifies a custom title for the current window.

SHOWTYPE <*entry-type* | ALL>
selects the type of catalog entry that is listed in catalog directories. By default, catalog directories list all entries in the current catalog. You can use the SHOWTYPE command to show only one type of entry in the listing.

Use the SHOWTYPE ALL command to restore the default behavior. To display the currently selected type on the window's message line, use the SHOWTYPE command without arguments.

TYPE <*entry-type*>
specifies the entry type that the procedure assumes when the type is not explicitly specified in a command. Each procedure that uses catalog entries has a different default type. You can use the TYPE command to change the default type to suit your needs.

To a display the current default type on the window's message line, use the TYPE command without arguments.

WSIZE
displays the size and position specifications of the current window. You can use the WSIZE command to determine the current window specifications before you execute a WDEF or WREGION command.

Concurrent Window Commands

You can open multiple windows from a single invocation of the FSLETTER or FSVIEW procedure. These windows are referred to as concurrent windows. The following commands are useful for managing concurrent windows in FSLETTER and FSVIEW sessions:

SWAP
moves the cursor to the next concurrent window and activates that window.

WREGION <*start-row start-column rows columns*>
WREGION CLEAR
specifies the area of the display that the next concurrent window that is opened by an EDIT or BROWSE command will occupy.

Note: The WREGION command affects only the next window that is opened, not all subsequent windows. △

Specify the starting row and column for the upper left corner of the new window and the window's size in rows and columns. You can use the global WSIZE command to determine the size of the current window.

Note: In character-based display environments, the area of the window that is available for procedure output is four lines and two rows less than the amount you specify in the *rows* and *columns* arguments. Remember that the top and bottom window borders, the command line, and the message line occupy a total of four rows, and the left and right window borders occupy two columns. For example, the following command would cause the next window to occupy the bottom half of a 24-line by 80-column display:

```
wregion 13 1 12 80
```

This restriction is not applicable to graphical windowing environments. △

Use the WREGION CLEAR command to clear any previous WREGION setting and return to the default layout in which concurrent windows partially overlap in a staggered pattern.

Use the WREGION command with no arguments to display the window's current region setting on the window's message line.

Printing Commands

FONT
opens the FONT window and displays the font control information from the current form. The listing shows what color and highlighting attributes the form interprets as signals to change printing characteristics. The FONT window is for browsing only; to change the font information you must open the FORM window and edit the FORM entry. For more information, see the description of the FORM window the online Help for Base SAS software.

Note: The FONT window is not opened if no font information is defined in the current form. Instead, a message is displayed indicating that the form contains no font information. △

FORMNAME <*form-name*>
FORMNAME CLEAR
specifies the default FORM entry, which contains instructions for printing images that are captured with the SPRINT command. The SAS System's default form is DEFAULT.FORM. You can use the FORMNAME command to specify a different default. For more information about forms, refer to the online Help for Base SAS software.

When a form is required, the SAS System looks for the specified FORM entry first in the current catalog (if a catalog is currently open). If the form is not there, the system next looks in your personal PROFILE catalog (SASUSER.PROFILE, or WORK.PROFILE if the SASUSER library is not defined) and then, finally, in the SASHELP.FSP catalog.

Use the FORMNAME CLEAR command to return to using DEFAULT.FORM as the default FORM entry name.

Use the FORMNAME command with no arguments to display the current default form on the window's message line.

PRTFILE <*fileref* | '*actual-filename*' <APPEND | REPLACE>>
PRTFILE CLEAR

specifies a file to which the procedure sends documents that are created with the FSLETTER procedure or with the SPRINT command. By default, output is sent to the printer destination that is specified in the current form. You can use the PRTFILE command to route the output to a file instead.

You identify the target file by using either a previously assigned fileref or the actual filename. If you specify a filename, it must be enclosed in quotes.

With the filename or fileref, you can also specify either the APPEND or REPLACE option to determine how output is handled when the file already exists. The default is REPLACE, which causes output sent to an existing file to overwrite the current contents of the file. To add the new output to any existing contents instead, use the APPEND option.

Use the PRTFILE CLEAR command to cancel the previous PRTFILE command and route output to the printer again.

Use the PRTFILE command with no arguments to display the name of the current print file on the window's message line.

SPRINT <FILE=*fileref* | '*actual-filename*'> <FORM=*form-name*> <NOBORDER>
SPRINT FREE

captures the contents of the current window (except for the command and message lines). Unless you use the NOBORDER option, the window borders are also included in the capture.

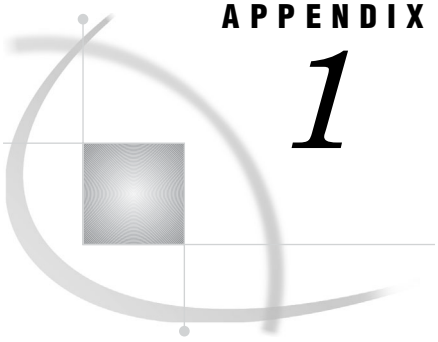
Output characteristics are determined by an associated FORM entry. You can use a FORMNAME command before issuing a SPRINT command to change the default FORM entry for all captures. Use the FORM= option with the SPRINT command to select a form other than the default for an individual capture.

By default, output is sent to the printer destination that is specified in the associated form. If you want to send all output to a file instead of to the printer, use the PRTFILE command before issuing a SPRINT command. To route an individual capture to a file instead of to the printer, or to a file other than the one that is specified by the PRTFILE command, use the FILE= option with the SPRINT command. Changing output destinations within a procedure automatically frees the previous print file or print queue.

Note: Once you have sent SPRINT output to a file, any additional output that you send to that file must use the same FORM entry. △

The print queue or print file that the SPRINT command uses is freed when you end the procedure from which you captured window contents. Use the SPRINT FREE command to free the print queue or print file before ending the procedure.

You can use the SPRINT command to capture information from several procedures in a single print file or print queue. In this case, the print file or print queue is not freed until you end all the procedures that sent output to the file or queue.



APPENDIX

1

Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *Base SAS Procedures Guide*
- *SAS Component Language: Reference*

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/pubs**
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

Glossary

attributes
the characteristics that describe and control the appearance and function of windows and window elements. See also field attributes, general attributes.

banner
the command prompt in the upper-left corner of a window.

computed variable
a variable in an FSVIEW window whose value is calculated by a formula.

control level
(1) the degree to which a procedure can control access to observations in a SAS data set. (2) one of the determinants in the kind of lock that a task obtains on a SAS data set or on an observation in the data set. The control level specifies how other SAS tasks can access the SAS data set concurrently. Every SAS task has an open mode and a default control level for each SAS data set that it accesses, based on how the task operates on that data set.

dialog box
a type of window that opens to prompt you for additional information or to ask you to confirm a request.

engine
a component of SAS software that reads from or writes to a file. Each engine enables SAS to access files that are in a particular format.

execution environment
the operating environment in which a user executes applications, as opposed to the development environment in which a developer builds applications. See also operating environment.

execution phase
the stage at which an SCL program is executed, from initialization through termination. During this phase, the program typically validates field values, calculates values for computed variables based on user input, invokes secondary windows, issues queries, executes user-issued commands, and retrieves values from SAS data sets or from external files.

execution stack
a last-in, first-out stack that lists the current and inactive entries that were called during the execution of a SAS/AF, FSEDIT, or FSVIEW application.

field
a window area in which users can view, enter, or modify a value.

field attributes
a set of characteristics that describe and control how the contents of a field are treated when values are entered or displayed in the field.

field validation
the process of checking user-entered values either against attributes that have been specified for a field or against conditions that have been specified in a SAS Component Language program.

fileref (file reference)
a name that is temporarily assigned to an external file or to an aggregate storage location such as a directory or folder. The fileref identifies the file or the storage location to SAS. See also libref.

form
a SAS catalog entry of type FORM. Forms contain information that is used for formatting printed output.

format
a pattern or set of instructions that SAS uses to determine how the values of a variable (or column) should be written or displayed. SAS provides a set of standard formats and also enables you to define your own formats.

formula
an expression that calculates the value for a computed variable or that manipulates the value of a data set variable in the FSVIEW window. Formulas can consist of more than one statement, and they can also include any of the elements of SAS Component Language that are valid in SAS/FSP software.

formula entry
a SAS catalog entry of type FORMULA that contains formulas for calculating computed variables, as well as other general parameters of an FSVIEW session. Only one formula entry at a time can be associated with a data set, but the associated formula entry can be changed or replaced during an FSVIEW session.

frame
a unit into which information that will be displayed in a window can be divided. Each frame consists of a block of text and fields that completely fill the window. For example, the FSEDIT Attribute window is divided into a series of fourteen frames–½one for each field attribute.

FSBROWSE application
an FSBROWSE session that has an associated SCREEN entry and which uses one or more of the following: a custom display layout, special field attributes, a SAS Component Language program, or custom window parameters.

FSEDIT application
an FSEDIT session that has an associated SCREEN entry and which uses one or more of the following: a custom display layout, special field attributes, a SAS Component Language program, or custom window parameters.

FSVIEW application
an FSVIEW session that has an associated FORMULA entry and which uses one or more of the following: a custom display layout, computed variables, special field attributes, or custom window parameters.

general attributes
the set of characteristics that are assigned to a SAS catalog entry (such as a PROGRAM entry or a FRAME entry) that displays a window.

global command
a command that is valid in all windows for a particular SAS software product.

initialization phase
in SAS Component Language, the stage at which initialization steps are performed before a window is displayed in SAS/AF software or before an observation is displayed in SAS/FSP software.

LETTER entry
a SAS catalog entry of type LETTER. A LETTER entry contains the text of a document that was prepared with the FSLETTER procedure. The LETTER entry also stores the attributes of any fields in the document, as well as the name of the associated form for printing the document and the general text editor parameters that were in effect when the document was last edited.

libref (library reference)
a name that is temporarily associated with a SAS data library. The complete name of a SAS file consists of two words, separated by a period. The libref, which is the first word, indicates the library. The second word is the name of the specific SAS file. For example, in VLIB.NEWBDAY, the libref VLIB tells SAS which library contains the file NEWBDAY. You assign a libref with a LIBNAME statement or with an operating system command.

menu bar
the primary list of items at the top of a window, which represent the actions or classes of actions that can be executed. Selecting an item executes an action, opens a pull-down menu, or opens a dialog box that requests additional information.

message area
the area immediately beneath a window's command line or menu bar which displays messages from SAS or from the SAS Component Language reserved variable _MSG_.

operating environment
a computer, or a logical partition of a computer, and the resources (such as an operating system and other software and hardware) that are available to the computer or partition.

parameter
(1) in SAS/AF and SAS/FSP applications, a window characteristic that can be controlled by the user. (2) in SAS Component Language (SCL), a value that is passed from one entry in an application to another. For example, in SAS/AF applications, parameters are passed between entries by using the CALL DISPLAY and ENTRY statements. (3) a unit of command syntax other than the keyword. For example, NAME=, TYPE=, and COLOR= are typical command parameters that can be either optional or required.

protected field
a field to which users cannot tab and in which they cannot alter values.

SAS catalog
a SAS file that stores many different kinds of information in smaller units called catalog entries. A single SAS catalog can contain several different types of catalog entries. See also SAS catalog entry.

SAS catalog entry
a separate storage unit within a SAS catalog. Each entry has an entry type that identifies its purpose to SAS. Some catalog entries contain system information such

as key definitions. Other catalog entries contain application information such as window definitions, Help windows, formats, informats, macros, or graphics output.

SAS Component Language (SCL)
See SCL (SAS Component Language).

SAS data file
a type of SAS data set that contains data values as well as descriptor information that is associated with the data. The descriptor information includes information such as the data types and lengths of the variables, as well as the name of the engine that was used to create the data. SAS data files are of member type DATA. See also SAS data set, SAS data view.

SAS data library
a collection of one or more SAS files that are recognized by SAS and which are referenced and stored as a unit. Each file is a member of the library.

SAS data set
a file whose contents are in one of the native SAS file formats. There are two types of SAS data sets: SAS data files and SAS data views. SAS data files contain data values in addition to descriptor information that is associated with the data. SAS data views contain only the descriptor information plus other information that is required for retrieving data values from other SAS data sets or from files whose contents are in other software vendors' file formats.

SAS data view
a type of SAS data set that retrieves data values from other files. A SAS data view contains only descriptor information such as the data types and lengths of the variables (columns), plus other information that is required for retrieving data values from other SAS data sets or from files that are stored in other software vendors' file formats. SAS data views are of member type VIEW.

SAS file
a specially structured file that is created, organized, and, optionally, maintained by SAS. A SAS file can be a SAS data set, a catalog, a stored program, or an access descriptor.

SAS text editor
a text-editing facility that is available in some windows of the SAS windowing environment, as well as in windows of SAS/AF, SAS/FSP, and SAS/GRAPH software.

SCL (SAS Component Language)
a programming language that is provided with SAS/AF and SAS/FSP software. You can use SCL for developing interactive applications that manipulate SAS data sets and external files; for displaying tables, menus, and selection lists; for generating SAS source code and submitting it to SAS for execution; and for generating code for execution by the host command processor.

screen
a division of an FSBROWSE or FSEDIT window. When a data set contains more variables than can fit in an FSBROWSE or FSEDIT window, or when the defined size of a custom display is larger than the current window dimensions, the FSBROWSE or FSEDIT procedure divides the information to be displayed into window-sized units called screens.

SCREEN entry
a SAS catalog entry of type SCREEN that contains information about the custom features of an FSBROWSE or FSEDIT application. If the application includes a SAS Component Language (SCL) program, the program is also stored in the SCREEN entry.

stream
a series of information about the entries and windows that are created when a user invokes a SAS/AF, FSEDIT, or FSVIEW application. Within the stream, SAS creates stacks to keep track of which entries have been called and which windows are currently open. Stacks use a last-in, first-out hierarchy.

termination phase
in SAS Component Language, the stage at which a program processes any steps that are required before the current observation is written to the data set and another observation is read.

validate field values
See field validation.

WHERE clause
one or more WHERE expressions used in a WHERE statement, a WHERE function, or a WHERE= data set option.

WHERE expression
a type of SAS expression that specifies a condition for selecting observations for processing by a DATA step or a PROC step. WHERE expressions can contain special operators that are not available in other SAS expressions. WHERE expressions can appear in a WHERE statement, a WHERE= data set option, a WHERE clause, or a WHERE command.

Index

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Your Turn

If you have comments or suggestions about *SAS/FSP® 9.1 Procedures Guide*, please send them to us on a photocopy of this page, or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: **yourturn@sas.com**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **suggest@sas.com**