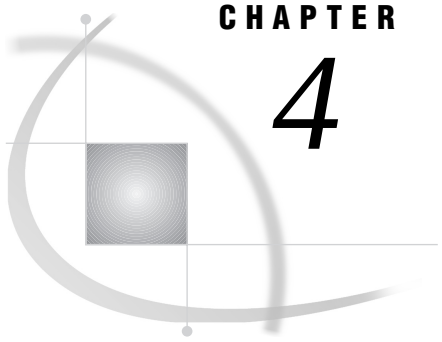


CHAPTER

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.

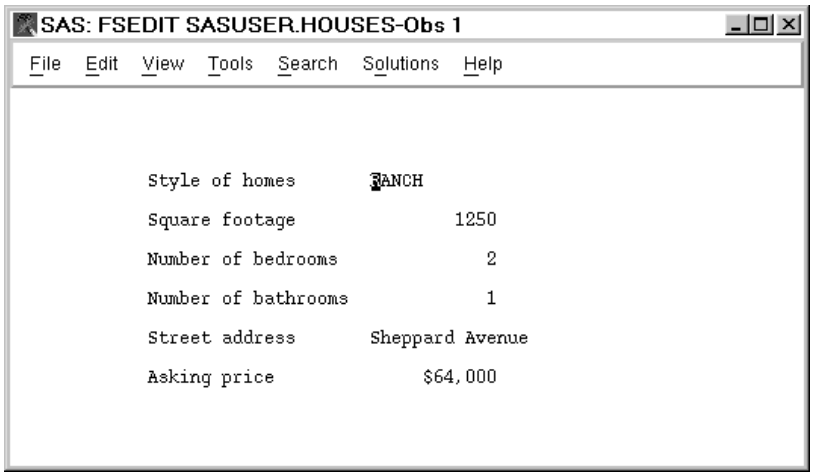
Note: Most of the features that are described in the following sections for the FSEDIT procedure are also available for the FSBROWSE procedure. However, the FSEDIT functions that relate to creating new data sets and to adding, editing, or deleting observations in existing data sets are not applicable 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 27 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 28 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 141, 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 65. 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 65 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.

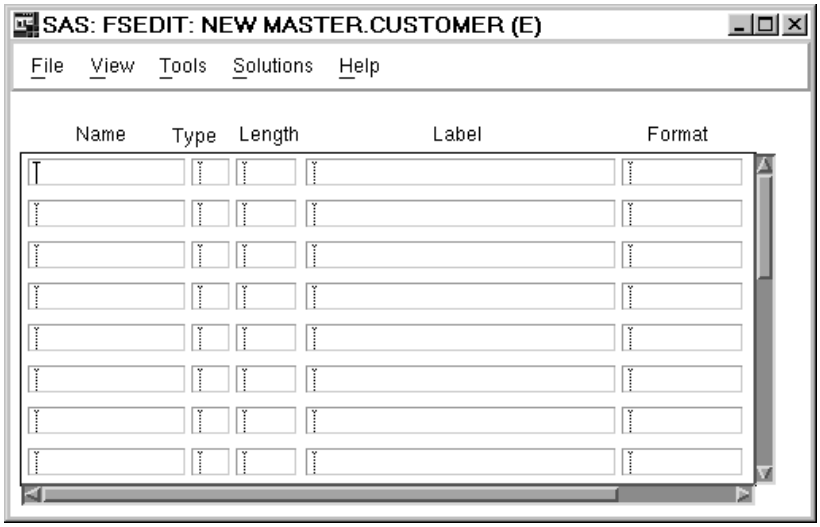
Opening the FSEDIT NEW Window

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 39 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 141, 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

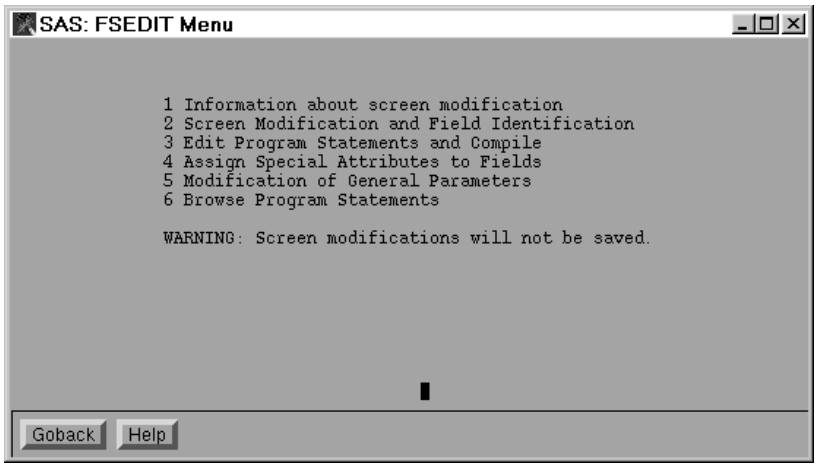
- the customized display format
- a SAS Component Language program
- attribute information for all of the fields
- the general parameters of the FSEDIT session.

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 43 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 44 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 53 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 53 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 58 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 53 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 58.

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
: 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.

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

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) ? _
```

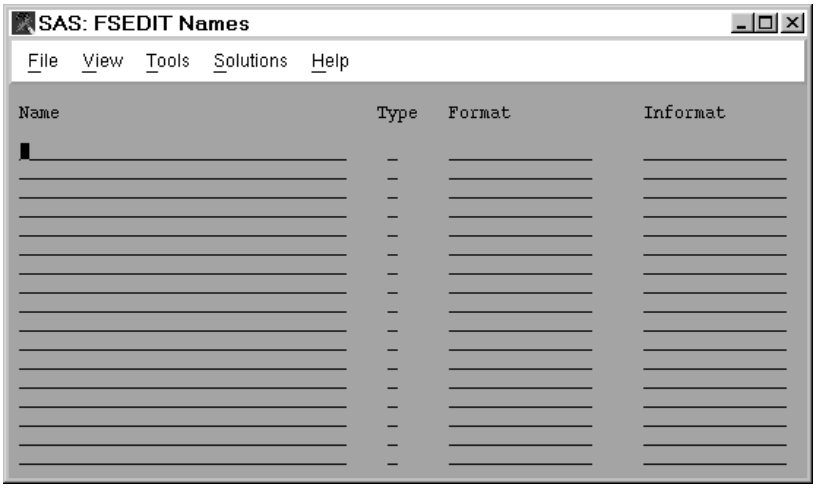
Your response determines whether you go directly to the field identification step or enter the field definition step first.

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 48 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 141, 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 28 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 141, 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 28 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. 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 53. 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 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 58 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 58 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 58 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. By default, all fields initially have this attribute when you create a new custom display. 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.

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 58.) 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 58.) 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 58.) 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 58.) 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

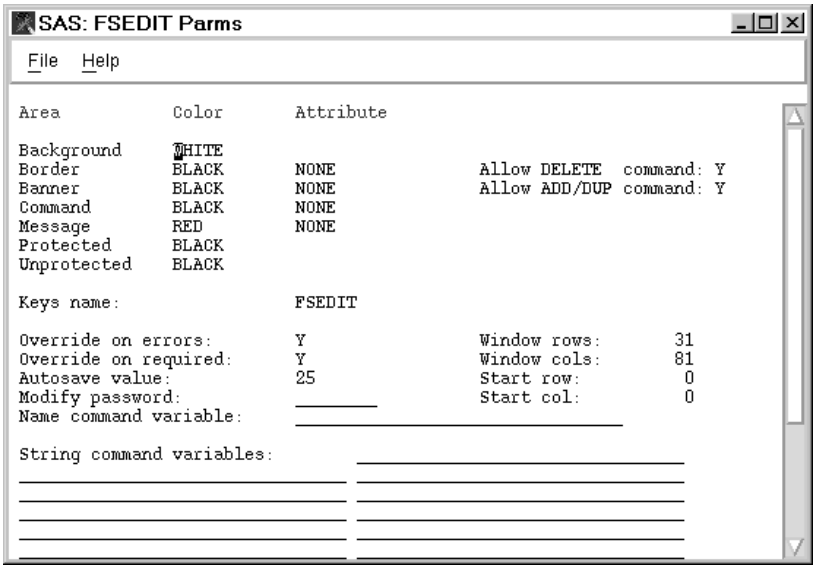
Most monochrome devices support only high intensity and underlining. If a user's device does not support the highlighting attributes that you specify, the highlighting attribute assignments are simply ignored. Therefore, you can assign these field attributes even though the application may not always 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 58 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 58 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.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/FSP® Software Procedures Guide, Version 8* , Cary, NC: SAS Institute Inc., 1999.

SAS/FSP® Software Procedures Guide, Version 8

ISBN 1–58025–517–5

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

1st printing, October 1999