

Developing SAS/AF® Applications for Reviewing Clinical Data

Carl R. Haske, Ph.D., STATPROBE, Inc., Ann Arbor, MI

ABSTRACT

Clinical review of data is a significant element of the data management process in the pharmaceutical industry. Conventional methods involve review of data using case report forms and hard copy of reports generated by a data management team and delivered to clinicians after the data management team finalizes the clinical database. Creative uses of computer technology allow clinicians real-time, interactive access to a database during the data entry and query process. Software systems that integrate the data management process and review process tend to deliver information faster, effectively increasing efficiency leading to accelerated production of a final database.

This paper discusses techniques for developing SAS/AF applications and presents a real-life example, STATPROBE DATA ACCESS®, an information system that allows clinicians and data managers to access and review data as it flows through a data management system. The system is easy to use due to a graphical user-interface incorporating SAS/AF frame entry tools. Object-oriented design makes the system flexible and facilitates the augmentation of custom requirements and features. The system exploits intrinsic SAS data structure attributes to drive the user-interface. Development has taken place on PC compatibles running OS/2® version 3 using SAS version 6.10. The code design will allow easy porting to other platforms.

INTRODUCTION

A important element in the clinical data management process is the review of the clinical database. Computer information systems assist the process by allowing the reviewer to quickly access and manipulate large sets of data and display only data of interest. Time savings are realized when the review and entry processes take place in parallel. Trends in the data can be noticed early and study protocols amended if necessary. The STATPROBE DATA ACCESS system discussed in this paper is a module to STATPROBE's data management system, STATPROBE DATAMAN®. The DATA ACCESS system allows ongoing access to data as it is being entered and managed, as well as after release of the final database.

The functional requirements for the system include the need to support multiple study protocols for any of STATPROBE's pharmaceutical clients. This requires the system to be general and 100% data-driven in order to accommodate database standards from many different companies. Using data set and variable labels to drive system radio and list boxes adds to the user appeal by avoiding cryptic names. The techniques used for this are described in the section titled Data Driven Dialogs and Look-Up Lists.

The system supplies two types of views, a single record view for detailed scrutiny of the data, and a multiple record view in spreadsheet format for more global investigations. Combining both types of views allows the user to obtain a clear "snapshot" of the data and look for trends over time. The

section Data Views describes how the two viewing procedures are implemented.

The system has a wide variety of users, many of who are not familiar with standard database concepts and the SAS system. Thus it is important that the system provide tools to help the user view the data. A primary aide in the system is a *search* facility that constructs SAS code via a set of intuitive dialogs that completely insulate the user from the actual SAS code. Searches are used to subset or restrict data. Searches are defined on individual data sets and may be saved to a search database for later recall. Saved searches can be modified and searches defined for one data set can be copied to a search for another data set. The section labeled Searching Data describes some techniques used to support the search facility.

Many users want highly customized features or reports on a database that must be programmed independently of the core system. However, the system architecture is designed to allow features and reports to be linked in and made accessible from a menu. The system architecture is described in the System Architecture section. The Custom Feature Examples section describes some examples of custom reports that have been integrated into the system.

Many users of the DATA ACCESS system are at a remote location, hence the data must be uploaded to their computer system. The system is set up to execute external procedures to accomplish data transfer with high speed modems. Details of this process are not discussed in this paper.

DATA DRIVEN DIALOGS AND LOOK-UP LISTS

Text items of all radio and list box objects in system dialogs are set dynamically when a frame is initialized. Information for initializing dialogs is captured from database attributes such as data set labels, variable names and labels, and variable values. This data driven aspect of the system interface is important from a developer perspective since each instance of the application runs on a different set of databases. Propelling menus, list selection boxes, and other selectable objects with data attributes avoids the tedious activity of re-defining selectable object with each new instance of the application.

The initialization vehicles are global SCL lists that store the database attributes. These global SCL lists are initialized at system startup. The global SCL lists are used for dialogs that have many objects to populate. Benchmark tests show this enhances system performance by decreasing fixed disk I/O and increasing response time. SCL lists are an superb data structure for storing the database attributes, however the developer must be aware that SCL list require more system resources.

The largest dialog in the system displays a radio box for project selection, two list boxes for data set and variable selection, two icons for task selection, two buttons for a custom list box item selection method, two buttons that link

to custom features, a "Transfer Data" icon, and a "Done" button to end the frame (see Figure 1).

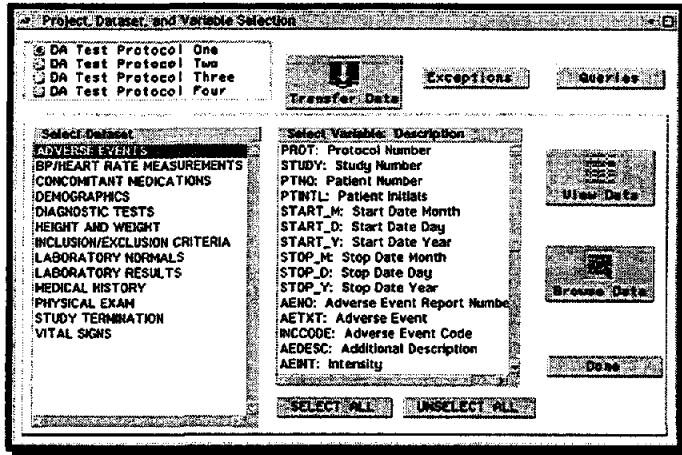


FIGURE 1. Project, Data set, and Variable Selection

The selectable projects are stored in a system data set that is modified for each new application instance. By default, the first project and the first data set for the project are selected at system startup. In addition, each time a project is selected, the data set list box is refreshed with the data sets for that project and the first data set is default selected. Likewise, each time a data set is selected, the variable list box is re-populated with the variables for that data set. All variables are selected as a default. The user selects only variables of interest to be used in the view procedures. The 'SELECT ALL' and 'UNSELECT ALL' buttons provide short cuts allowing the user to toggle full variable selection.

Each project item comprises a different study protocol. The project radio box and two list boxes use named SCL lists for population. The project selected determines the SCL list used to populate the data set list box and selection of a data set determines the SCL list used to populate the variable selection. This dependency is illustrated in Diagram 1. Each project, data set, and variable are assigned an index. These indexes are used to access the information in the SCL list. When a user selects a project and data set, global macro variables are set to store the corresponding indexes.

The project SCL list consists of list items of character strings corresponding to project titles. These project titles populate the project selection radio box. The data set list box is supported with an SCL list, indexed by the project number, with each item a named SCL list of data set names and labels. The data set labels are used to populate the data set list box. The variable list box is supported by the most complicated SCL list structure. It is a tree structure with three levels of branching and each node is an SCL list. The tree is indexed by project and data set number. The nodes at the lowest level of the tree are named SCL lists containing variable names and labels. Both variable names and labels are used to populate the variable list box.

Generic methods support initializing and accessing data from these SCL list structures. For example, suppose the user is viewing the j^{th} data set of the i^{th} project and wants to display a list of all variables in the data set. The user activates a control, and SCL code executes passing (i,j) to a method that accesses the SCL tree and returns a list id for the SCL list of variable names and labels for the j^{th} data set of the i^{th} project. This SCL list id is passed as an entry parameter to a universal frame containing a list box object that is populated by the SCL list. These techniques make it possible to maintain

standard utility frames usable by all the major functions in the system.

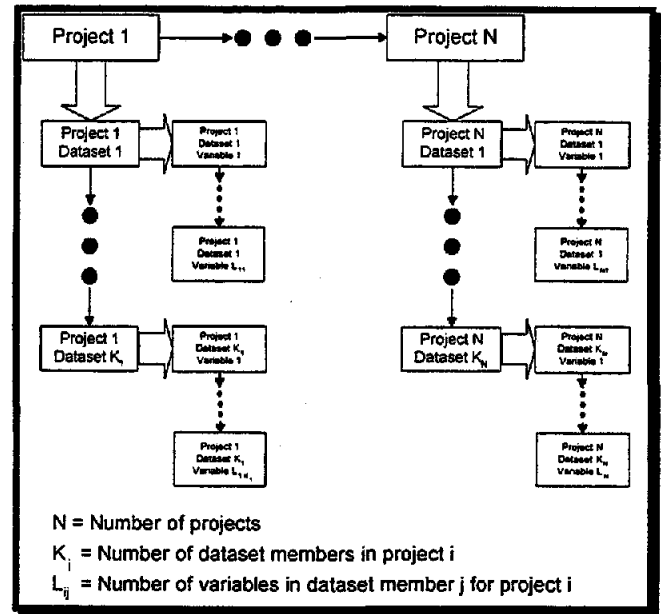


DIAGRAM 1. SCL list structures supporting project, data set, and variable selection objects

DATA VIEWS

Two facilities are provided for examining the data. The browse facility uses the CALL FSEdit function to initiate an FSBROWSE session using variable labels to display data in a default screen. The user is able to examine the data one record at a time at a highly detailed level. The standard SAS pmenu was used for the FSBROWSE sessions providing the user with a user friendly way to search through the data.

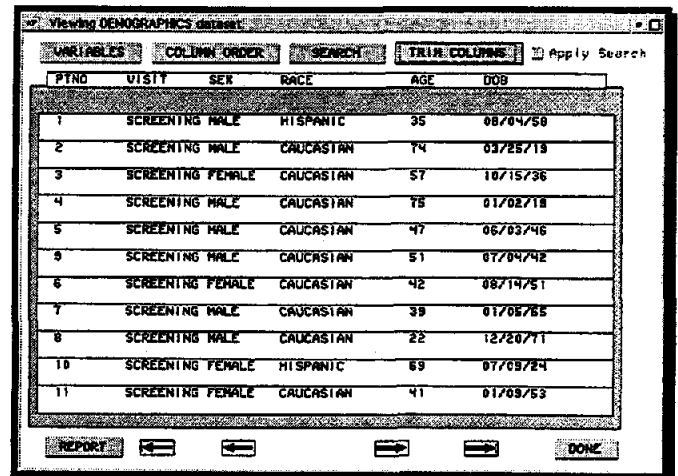


FIGURE 2. Viewing data in spreadsheet format

The view facility utilizes an extended table to display data in spreadsheet format (see Figure 2). In programming the extended table, several things were taken into consideration. In general, variable labels are too long to fit in the table header. Therefore the system displays actual variable names. If a variable has a format stored in the data set, the format is used to decode the data for display rather than displaying the actual data value. The column widths are set by the lengths

of a variable's data and, if available, a variable's format and are stored in an SCL list for use in scrolling the table as described below. The initial ordering of the columns uses the variable order in the data set.

Buttons to help the user navigate data are placed below the table. Controls are programmed to horizontally scroll the table by one column per execution left or right. This is done because the extended table's built-in horizontal scroll bar does not scroll the table in a manner natural for spreadsheets. The scroll amount for "Fast-Forward" and "Reverse" controls are programmed to right-align and left-align the columns. The extended table's built-in vertical scroll bar controls vertical scrolling of the table. A scroll bar is programmed to view data in columns wider than the display of the extended table. This scroll bar is kept hidden and appears below the extended table only when needed. This scroll bar scrolls the text field one character at a time.

Controls to help understand and modify the view are above the extended table. The "VARIABLES" button pops up a cross reference of variable names and labels to aid the user in understanding cryptic variable names in the table header.

The "COLUMN ORDER" button shows a list of column fields and allows the user to reorder the column display (see Figure 3). The column order is stored in an SCL list that acts as an index map. The default index map, defined when the frame initializes, is ordered by the variable numbers in the data set. The user can return to the default ordering at any time by clicking the "DEFAULT" button. The "SORT" button sorts the variables alphabetically. The "RESTART" button returns the variables to the order at the beginning of the column re-order operation.

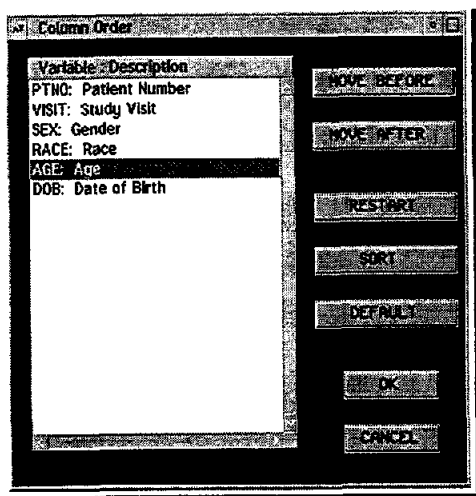


FIGURE 3. Column re-ordering

The "SEARCH" button initiates the search utility described in the next section. The "TRIM COLUMNS" button scans data for the value with the maximum width and re-sizes the column to that width. The "Apply Search" check box toggles the application of a search.

A "REPORT" button displays a report of the data in the SAS output screen where the user can choose to print it from the standard menu.

SEARCHING DATA

The search facility provides a way for the user to focus on different types of data and look for trends in data. The

approach to searching the data is by comparison of variables to a single value, a range of values, a list of values, or null. Searches are user defined on each data set and can be stored in the search database with a unique name. Once a search has been defined and stored it can be recalled at a later time then modified, copied for application to a different data set, or deleted.

There are eight dialog frames to help the user construct searches that query and subset the data. When the user executes the search facility, the first screen they see is a list of searches that have been previously defined and stored (see figure 4). The list is empty if no searches have been defined for a data set.

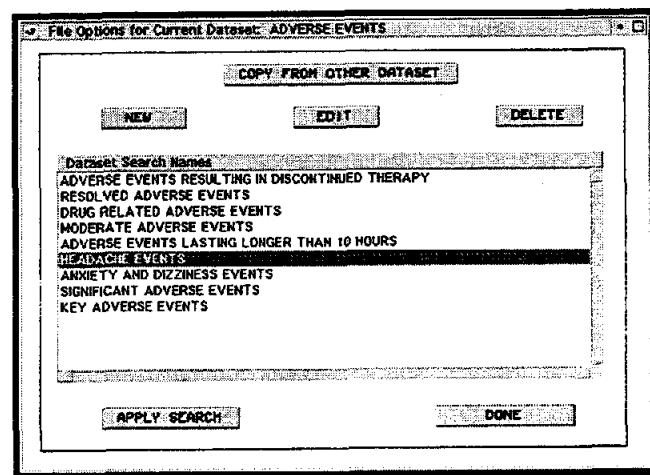


FIGURE 4. Selecting a saved search

The user can select single or multiple searches and apply them. Other options are to define a new search, edit a single search, or delete single or multiple searches.

When searches are selected and applied, only data that satisfies the criteria specified in all selected searches will be display. Thus multiple searches are logically ANDed together and applied to the data set. The user can select a single search for editing. After an existing search is edited, it can be saved over the existing search or saved to a new search.

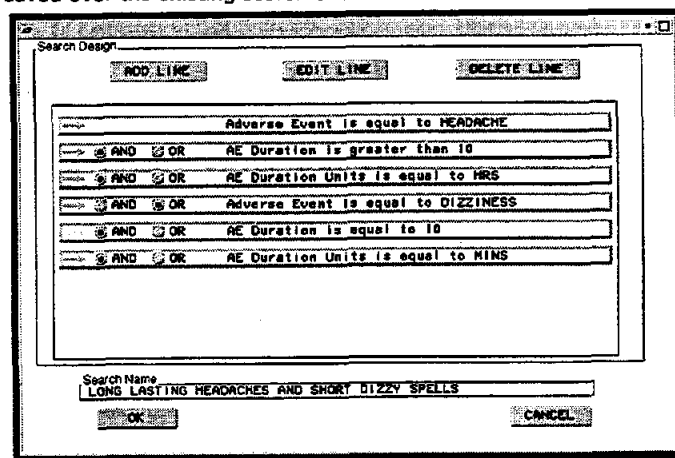


FIGURE 5. Designing a search

A single search consists of a collection of conditions, each on one variable (see Figure 5). These conditions are connected via logical ANDs and ORs selected by the user. Complex

searches are constructed by applying multiple simple searches.

To define a new search, a list of variables in the data set is displayed (see figure 6) to specify the first line of the search. The user selects the key variable to define the search on and one of the four types of searches they wish to define. Types of searches are direct comparison, restricting to a range, restricting to a list of values, and comparing to null. Then the user enters a dialog to define a single line of the search.

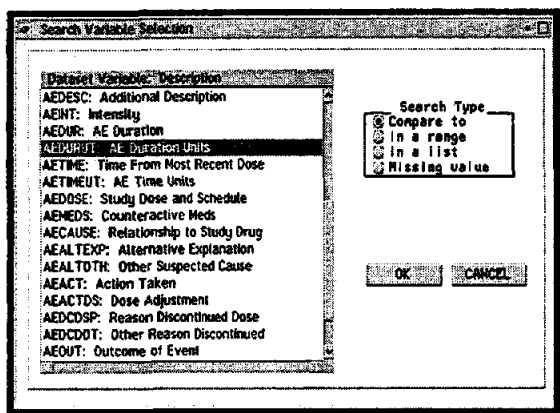


FIGURE 6. Search variable selection

As the search line is constructed, the English equivalent appears in a text box at the bottom of the dialog box (see figure 7). English text strings are constructed by the system to represent the search condition. Data set values of the variables are optionally displayed in a list box to help the user make decisions while constructing the search. For coded data, the formatted values are displayed in the list box. The user can select from the list of data set values to specify values to include in the search. Specific values can also be entered directly. Comparison to other variables in the data set can be included in the search by selecting from a pop-up list of data set variables.

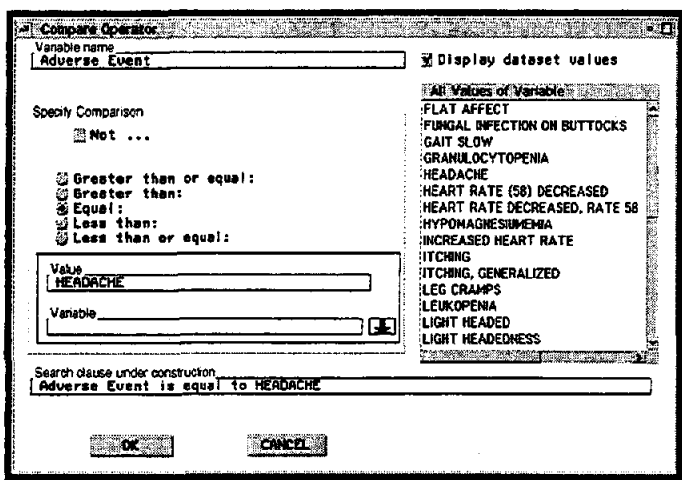


FIGURE 7. Constructing one line of a single search

After completing the line and clicking the 'OK' button, a frame that lists all the lines for the search is displayed in an extended table (see figure 5). A new line is added by again showing the user a list of variables to select from, requiring the user to select the type of search, and construct the new search line. The lines of the search are connected by selecting

logical ANDs and ORs represented in a radio box preceding each line. Once all search lines have been constructed, the search is named and saved permanently to the search database.

The search database is relational, consisting of six SAS data sets (see diagram 2). Programming tools were developed to manage the search database. When a search is applied, the search database is accessed and SAS code for a succession of where clauses is constructed and executed. Users find the search facility particularly helpful. An early, less sophisticated prototype required the user to understand proper SAS syntax in order to construct a where clause. The search facility in this system is much more sophisticated and completely insulates the user from SAS programming code.

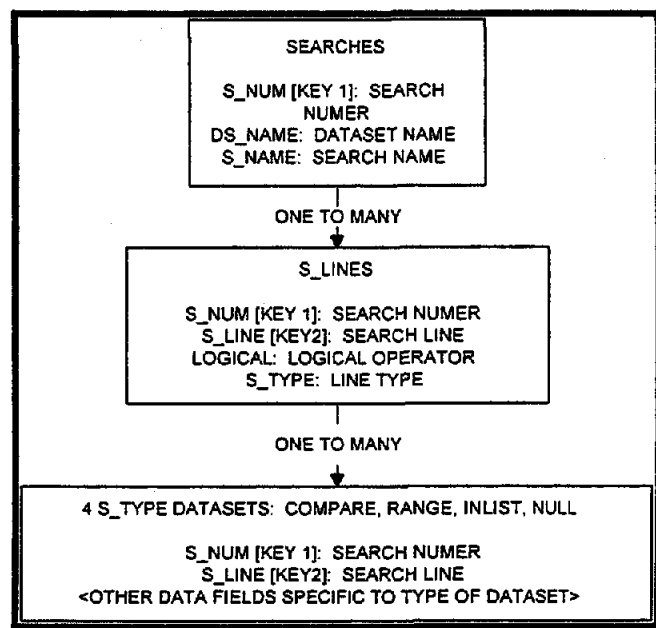


DIAGRAM 2. Search database (relational)

SYSTEM ARCHITECTURE

The DATA ACCESS application was designed to be customizable for use by many clients. This required a logical and physical separation of the core functionality and the features that are added for customization. To achieve this, the core of the system is stored in its own SAS catalog referred to as the kernel. The current version of the kernel has 43 frame and SCL entries. Maintenance and further development can occur on the kernel independent of various custom implementations of the system.

To build a production system another SAS catalog with frame entries supporting the custom features is developed and merged with the kernel. Three standard SCL entries containing code that varies with each implementation are always part of the custom catalog. The library entry contains a method that defines library references needed for the system to execute. This method is called only once at system initialization.

A second entry called 'custom.scl' contains methods that interface the kernel to the custom features and definitions for the button labels that invoke the custom features. Note the custom buttons labeled 'Exceptions' and 'Queries' in the example displayed in figure 1. These buttons' labels are stored in global macro variables and assigned in the INIT

section of the main dialog frame. If a button label is blank, the button is hidden.

A third entry is called 'download.scl'. This entry contains a method that is called when the 'Transfer Data' icon is selected (see figure 1). This method normally makes a system call to an application that performs a remote data transfer via a modem.

The custom catalog typically contains additional entries to support customized features that have been developed according to specifications detailed by joint work between STATPROBE's data management group and the client.

CUSTOM FEATURE EXAMPLES

The object-oriented design of the system makes it flexible enough to link in custom programming for each project. Many users want to see specialized exception reports, queries, data reports, or statistical reports. The system allows for these in the main dialog (see figure 1). Custom features are implemented by either saving reports as SAS data sets with descriptive labels or developing a set of programs and a program database to drive a menu.

In the case where reports are saved as data sets, when the user chooses to see a report, the report database is accessed and a menu of reports is constructed using the data set labels (see Figure 8). The user can use the viewing facility to examine the report. All of the features of the viewing facility are available, including the search facility. The user can choose to have the system print data on the report by selecting the "REPORT" button. The results are sent to the output screen where the user can choose to print a hard copy.

In the case where a custom set of programs is developed, the program database is accessed to build a menu of reports. When a specific report is selected, the corresponding report program is submitted resulting in output for the user to view.

Additional types of custom reports can easily be developed and linked into the system since all the programming tools are modular and accept the type of report and report database's physical location as parameters.

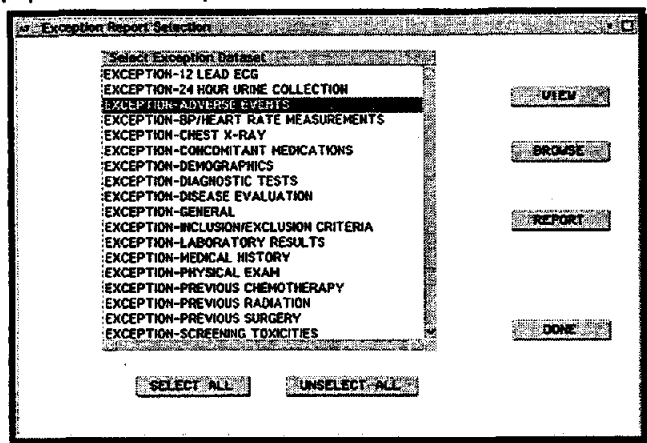


FIGURE 8. Choosing an Exception Report

SUMMARY

The system has been received positively by our clients and the base of clients using the system continues to grow. We are currently developing enhancements beyond version 1.5 of the DATA ACCESS system. Many features are being designed

into the next version of the software including saving views of data, posting notes on the custom reports, cross tabulation queries, and message posting capabilities. Other modules to be developed and linked into our data management system are reporting capabilities for summary statistics computed on the data.

The SAS/AF Frame entries are a good development platform for software. The built in tools allow for quick prototype development and give the application developer the freedom to try many different approaches to the software in a short amount of time. The abilities to define derived classes and override methods is a necessity for serious application development.

An initial prototype for DATA ACCESS was developed in less than two weeks. At this point we had most of the user-interface screens. A working prototype followed after an additional six weeks. Version 1.0 was launched four months later. Adding the search feature and maintenance fixes brought about the release of version 1.5. We anticipate a few more intermediate releases for maintenance and minor enhancements prior to delivering version 2.0.

REFERENCES

SAS Institute, Inc. (1993), *SAS/AF Software: FRAME Entry, Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

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

SAS Institute, Inc. (1991), *SAS Technical Report P-216: SAS/AF Software, SAS/FSP Software, and SAS Screen Control Language: Changes and Enhancements, Release 6.07*, Cary, NC: SAS Institute Inc.

Haske, Carl (1995), *Using SAS/AF and Frame Entry to Access Data*, SUGI 20 Proceedings, Cary, NC SAS Institute Inc.

ACKNOWLEDGMENTS

SAS and SAS/AF are registered trademarks of SAS Institute Inc. in the USA and other countries. IBM and OS/2 are registered trademarks of International Business Machines Corporation. ® indicates USA registration.

Thanks to Matthew Becker, Mary Fleischer, Roger Gibb, Terri Madison, D.J. Penix, Kelly Robbins, Jennifer Savage, Jennifer Sievers, and Peter Slintak for their contributions to the development of STATPROBE DATA ACCESS.

AUTHORS ADDRESS

Carl Haske, Ph.D.
STATPROBE, Inc.
3885 Research Park Drive
Ann Arbor, MI 48108
(313) 769-5000 x115
E-Mail:
• 72700,1075@compuserve.com
• statprob@aeonline.com