Electronic Data Processing dates back to the 1890 United States Census when Herman Hollerith reversed the idea employed in Jacquard’s loom to count people. Both devices were a response to the need to handle a large number of similar operations systematically and quickly. The loom was used to combine diverse colors into complex tapestry; Hollerith was, if you will, unravelling the social tapestry of the United States and recording where what was where.

Well into the middle of the twentieth century data processing methodology had developed little from the turn of the century. Those who remember unit record equipment will confirm that they were more mechanical than electronic. Programming, when it occurred, was carried out by connecting terminals on a removable board with a set of copper wires. If the frustration of “programming” such a board seems daunting, consider the data source—punched cards, which were victims of humidity, errant coffee, folding, stapling and mutilation. We used to receive all kinds of bills on these eighty column wonders, with the appropriate warning. Every billpayer felt somehow de-humanized upon receiving such products of automation. The real sympathy should have gone to the poor data clerk who received them and then had to run them through unit record equipment. For, if the volume of data was sufficient to justify their use, then the data handling problems increased in geometric proportions to the savings. Anyone old enough to have used a sorting machine can verify the horrible feeling that a card jam between the 6 and 7 slots can bring on.

We have come a long way from those old days: sort programs don’t jam; the art of wiring boards is lost, replaced by programming languages. It didn’t take long after the introduction of the language compilers for parameter driver programs to be developed. The parameter driven programs were not very friendly, one had to know what went in columns 2-6 and had best not make a mistake anywhere, but at least one did not have to write a new program for each analysis of variance problem. With these programs the computer user expanded beyond the nimble-fingered and nimble-minded programmer.

The introduction of SAS® was a revolutionary step forward in the development of user-driven software. With the concept of the SAS® dataset and access to very sophisticated procedures, the user had a dramatic increase in computing power available to him. The ease of using the DATA step is probably only appreciated by old programmers who struggled with uninitialized variables, incomplete dimension statements, etc.

Ironically, today it is the DATA step that causes the most grief to the average SAS® user. The division between the many possible ways is which data can be defined, and the discovery, at some point after one has settled on a definition, that the relationships established don’t really suit the user’s needs can lead to complete user frustration. The popularity and extensive use of PC spreadsheet programs such as LOTUS 1-2-3 shows the direction of future user-driven software development. With the spreadsheet program logic, all operations and data are contained within the same x-y plane. Their success has come about because they have effectively closed down the "space" in which a user can function. No matter when one looks at a spreadsheet, it consists of a finite set of rows and columns. The user is forced to physically place a data element in its desired position. With SAS® all of the relationships are logical, and as such carry a greater freedom and require a greater understanding of data structures.
A Limited Menu System

While the advent of spreadsheet programs has increased the use of computer technology and spread it into more hands, those of us who are involved in the maintenance of large databases for corporate data reporting and analysis find that SAS® provides a set of sophisticated and elegant tools that allow for complexity and do not require extreme levels of sophistication for novice users. The actual operation of such systems -- the submission of jobs, creating standard reports, etc. -- are the responsibility of people with minimum computing skills. To allow these people to operate comfortably, we have developed a menu driven system that takes advantage of SAS/AF®, SAS/FSp® and macro processing. In our development the primary goal is to eventually make every operation seem as though it is a function of user-controlled data.

To do this, we established a general structure where the user gains access through the following three gateways:

1. The system is initialized by the execution of a parameter setting macro which establishes defaults for the session.

2. PROC DISPLAY is then invoked, accessing a centralized menu. SAS/AF programs and macros are invoked by user choice.

3. PROC FSEDIT is generally used to supplement PROC DISPLAY for more complex data manipulation.

By using these features of SAS® we can establish control of the environment and provide a comfortable "space" within which the user can work with ease and efficiency.

These operational features are made possible by using three special types of datasets that are created under the control of the system:

MACRO PARAMETER: The macro parameter dataset is a set of paired variables that are used for the length of a session. In our system, datasets and variables are named on the basis of these parameters. In addition, key variables that are used repetitively in calculations, and with which the user may wish to experiment are also maintained in this file. By selecting a choice on an AF screen the user invokes FSEDIT to modify or review these parameters.

DIRECTORY: All of our major databases have simple dataset directories. These contain a brief description of the history of the members. The directory contains only information on members that were created under control of the integral components of the system. That is, a user may create a new member for his own use which will not be included in the directory, but all necessary members are contained here. This dataset is used for general housekeeping purposes such as backup and restores and also is a source of information for many menu-driven operations. It is perhaps the simplest of ideas that pays off with the greatest dividends when you have a wealth of data with which to deal.

POINTER: For large master datasets we have developed a system of pointer locations. Briefly, the pointer dataset is used for rapid access to elements within this dataset, so that the user works upon a subset of the master file. The pointer dataset has the relationship of the individual data item to an organizational group or resource. For example, given a general ledger system the pointer dataset, in addition to giving an element's location in the master file, would show the element to be a particular account which was a liability or asset for Division X. The maximum number of reads required for any data element is never greater than three.
The system, as we described it, is closed in so far as the functions that are provided are those that are given on the AF menu and program screens. While it does provide for a comfortable working space for those who have only a specific set of functions to carry out, it is little comfort for those who wish to order something off the menu. As the user base for the data grows the variety of demands also grow. Instead of placing more and more customized items on menus it is obvious that we must follow the lead of the fastfood industry and offer up a "salad bar" of data for our customers, and provide them with the utensils to get it digested.

**The TOOLBOX Approach**

Using SAS/SHARE® we are now able to provide to a very large user population a comprehensive set of business related data. However, the user who is unfamiliar with SAS® would have a difficult time making use of such data if he is confronted with the infinite range as his first sample. It is for this reason that we are in the process of developing for the user a method whereby they can begin with small sets of data in a controlled space. We have even bowed to the PC market by giving our system the name "ToolBox". The principle idea is to provide them with the power SAS® has to offer with the minimum of education. To do this we have forced ourselves some better habits in using the self-documentation facilities available in SAS® such as labels for datasets and variables, explicitly defined formats, etc.

Access to the TOOLBOX system is gained by issuing the TOOLBOX command in TSO. At this time the user will have access to whatever databases that he has authority to via the computing system's security system. He also has the option of nominating his own SAS® databases. The TOOLBOX system consists of a SAS® dataset containing as AF catalog, FS EDIT screen catalog, an OS macro library and a SAS® format library. The user is required to know SAS® library names (LIBNAME), and the OS dataset name if he has his own SAS® database. If he is using the TOOLBOX on his own database, the system will generate for him a DIRECTORY member. This member is required for subsequent operations using the tools. From this time on the system is menu driven. For an example here we will consider PROC TABULATE.

**PROC TABULATE**

PROC TABULATE is one of SAS's most valuable procedures for presenting data. It is also one of its most unpredictable in terms of coding the required tables statements compared with the end results. After selecting the appropriate option on the menu, the user is presented with the option of either selecting a table defined previously or create a new table. If he opts for a new table he is then presented with FS EDIT screen containing a list of the variables available in the active library. After making this selection, if the member contains more than 20 variables, he is presented with another FS EDIT screen containing a list of the variables available in the member. He is then presented with an AF screen which attemptes to give him an idea of the physical location of the variables in the output listing, along with variables available to him. He must fill in the blanks. After completing this selection process, he is presented with another AF screen which allows him to spruce up the labelling of the output. Finally he is required to designate the output destination. After the procedure has been performed he is asked if he would like to keep the procedure for later use or documentation purposes. If so, the data is stored in an OS PDS with a name he provides.
The internal steps are these:

1. PROC FSEDIT the Directory member - For member selection. (OPTIONAL).
2. PROC CONTENTS on the selected member. (OPTIONAL)
3. PROC FSEDIT on the contents output dataset. (OPTIONAL)
4. PROC DISPLAY and associated Macros for the table definition.
5. Fast branch issued recursively based on the number of analysis variables chosen.
6. Fast branch to set the labeling information.
7. Fast branch to obtain the output destination.
8. A DATA step to generate the program code to a temporary OS dataset.
9. Finally, PROC TABULATE.

Those of us who remember the days of unit record equipment, would consider these hidden steps as so much overhead, and might reasonably ask if it was all worth it. The short answer is yes. By providing the novice user with access to the power of SAS in this form is justified, if for no other reason, that it may make him more efficient in his analysis of the data. Given the raw equation of 1 PROC TABULATE against PROC TABULATE plus 1 to 8 above, obviously the former is more efficient in terms of computing resources, but if one considers that he might not even approach the problem without access to the TOOLBOX, then the intangibles, such as lost revenue, lack of information in decision making, etc., have to be considered. Who knows, the user may get tired of beating his way through all of those menus and decide to code it himself.

Conclusion

The two directions that we have taken in our development of a Management Information System do not differ in any great detail. Our operational staff can quickly and effectively do their jobs and our programming staff can concentrate on the development of tools for users rather than one-off reports. The use of SAS® -- as a tool to speed along at development and to provide that comprehensive set of tools for our users to perform data analysis -- is the key to success.

NOTE

Because of the fact that the procedures described here rely extensively on macro processing containing macro variables called from AF screens which may be called from other macros, no samples of programming are provided. For those interested in sharing in the development of TOOLBOX, the author will gladly provide copies of the programs on a diskette of your choice, along with documentation.

The programming for the TOOLBOX menu system was done by Andrew McIntryre, Group Management Information Services, BNZ.

SAS, SAS/GRAPH, SAS/SHARE, and SAS/FSP are registered trademarks of SAS Institute Inc., Cary, NC, USA.