While SAS is well known for its ability to save programmers both time and effort, certain applications remain tedious and time-consuming even after SAS is employed. One such application which has frequently been encountered at BTI has necessitated development of a program which is able to generate SAS code for use in routine situations. A description of the need for and development of the SASGEN program follows.

THE PROBLEM

After data have been collected and converted to machine-readable form, project leaders frequently desire a "picture" of the data, usually in the form of frequencies (or descriptive statistics) for each variable in the dataset. In such situations, PROC UNIVARIATE or PROC FREQ is a simple way to obtain the desired distributions with a minimum of programmer effort. However, when the number of variables to be analyzed is very large, writing and entering the SAS code to perform the analysis can become a very time-consuming task. The problem is further complicated if recodes are to be done, or if variable labels are desired. As such as datasets containing upwards of 500 variables are routinely encountered at BTI, it became evident that the time devoted to summarizing the data could approach that spent collecting it.

Survey datasets are generally accompanied by a machine-readable codebook (or dictionary) containing the name of each variable and its location in the file. In many cases, the codebook will also specify variable types (character, real, integer) and provide variable labels. In other words, if the user has a codebook, he/she will have, in machine-readable form, most of the information necessary to create a SAS file and proceed with an analysis of each variable.

Consider the simple codebook in Figure 1 below. The codebook contains variable names, beginning and ending locations in the data file, and variable labels. Variables containing non-numeric characters are denoted by an 'A' to the right of the variable name. To perform analysis using SAS, one could, using an interactive editor, construct an INPUT statement by deleting the variable labels from each line, inserting a hyphen between each beginning and ending location, converting each 'A' to a ';' placing a semicolon after the ending location for the last variable, and inserting the word 'INPUT' before the first variable name.

| FIGURE 1 |
| A Simple Codebook |
| IDNO 01 06 ID NUMBER |
| AGE 07 08 R'S AGE (YEARS) |
| SEX A 09 10 R'S SEX (M OR F) |
| RACE 11 12 R'S RACE |
| EDUC 13 14 EDUCATION LEVEL |
| INCOME 15 20 ANNUAL INCOME |

In addition, a LABEL statement can be created by deleting the column locations and replacing them with '=' signs. By a similar process of replacement and change, the appropriate PROC statement can be generated.

While this approach may save substantial time and effort over the hand keying of all SAS program statements, it still remains a cumbersome process if a large number of variables is involved. In addition, the structure of codebooks is not always compatible with the construction of SAS statements. For example, codebook entries for alphabetic variables may have the flag for alphanumeric variables after the column locations, a fact which would complicate the creation of an INPUT statement.

THE SOLUTION

Inasmuch as the information needed to create a SAS dataset and run appropriate analyses was contained either in the codebook itself (variable locations and attributes) or in the specifications for analysis (e.g., treatment of missing values, procedures to be used), it became advantageous to develop software which could combine codebook information and user options to generate the appropriate SAS code.

What SASGEN Does

SASGEN uses as input a machine-readable codebook and a set of user-supplied parameters, and produces a file containing the SAS program used to produce frequencies and/or descriptive statistics on all variables in the data file described by the codebook. The complexity of the
SAS code generated by the program will depend upon the parameters supplied by the user. At its simplest, the user need not supply any parameters, and the program will assume various "defaults" concerning the structure of the codebook and generate only INPUT, LENGTH, and PROC UNIVARIATE or PROC FREQ statements. If the user so specifies, the generated SAS program is capable of such tasks as processing segmented files, producing variable labels, deleting certain variables, and recording 90-series codes (90-59, 990-999, etc.) to missing values. In general, the output SAS program is designed to generate a PROC UNIVARIATE statement to cover numeric variables, with the "FREQ" option used for variables having a field width less than a threshold specified by the user. PROC FREQ statements are generated for character variables.

In addition to creating an output file containing SAS statements, SASGEN produces a diagnostic and informational report, including a list of all parameters used in the generation of the statements, whether explicitly supplied by the user or employed by default. If any errors are encountered in the processing of parameter cards, SASGEN aborts and produces an error report.

The Codebook

The codebook must contain one record for each variable in the file, and each record must contain the variable name and either the variable's location in the file or its field width. So long as these conditions are satisfied, a great deal of flexibility is permitted insofar as the actual format of the codebook record is concerned. For example, in addition to processing single fixed-length data records, SASGEN can write code to process card-image records or segmented files; thus the codebook may contain information about the card or segment associated with each variable. In addition to making use of variable names and locations, SASGEN can also employ variable labels and flags for alphanumeric variables.

User-Supplied Parameters

User-supplied parameters serve two functions: (1) to describe the codebook and (2) to specify what the generated SAS program is to do. Each parameter is supplied on a separate record, with a keyword beginning in column 1 and parameter values (where appropriate) in a free-field format, separated from the keyword by at least one space.

The following are some parameters which can be used to describe the format of the codebook:

- NAME <v1> <v2> Beginning & ending location in codebook of variable name.
- BC <v1> <v2> Location in codebook of the value for the beginning location of the variable in the dataset.
- EC <v1> <v2> Location in codebook of the value for the ending location of the variable in the dataset.
- WIDTH <v1> <v2> Location in codebook of the field width of the variable. (To be used only if BC and EC params are not present.)
- ALPHA <v1> <v2> Location in codebook of the character(s) used to signify a character variable.
- ALKEY <character(s)> Characters(s) in codebook which signify a character variable. The codebook field specified by the ALPHA parameter will be scanned for the characters specified on this card.
- LABEL <v1> <v2> Location in codebook of variable label.
- FORWARD If the codebook field for the variable name is greater than 8 bytes, the first 8 bytes are to be used as the variable name. Otherwise, the last 8 bytes will be used.
- SBGC <v1> <v2> (For segmented data files) Codebook locations of the segment numbers.
- SEGD <v1> <v2> The beginning and ending locations in the data records of the segment number.
- CARD <v1> <v2> (For card-image data files) The location in the codebook of the card on which the variable is located.
- MAXN <v1> The maximum field width of numeric variables for which the "FREQ" option is to be
used with PROC UNIVARIATE. Numeric variables of width greater than 'v1' will have UNIVARIATE run without the 'FREQ' option.

MAXA <v1> The maximum field width of character variables to be included in PROC FREQ statements. If its field width is greater than 'v1', the variable will not be included in frequency tables.

MISSING If this parameter is present, numeric variables will have 90-series codes converted to missing values. 'v1' will be recoded to .A, 92 to .B, and so on.

SKIP <v1> The first 'v1' variables in the codebook will be dropped from the SAS dataset. This option is useful when the first few variables contain ID information.

DUPCHECK Causes the SASGEN program to search for duplicate variable names and change any by either adding a two digit sequence number to the end of the variable name or dropping the last two characters in the name and replacing them with the sequence number.

As noted earlier, many of the parameters have default values that are employed if the user does not supply the appropriate cards. Thus, it is possible for parameter cards to be completely omitted. In such a case, SASGEN assumes that the codebook contains variable names in columns 1-8; that beginning and ending field locations (instead of field widths) are used, and are located in columns 15-18 and 20-23, respectively; that neither character variables or value labels are present; that the 'FREQ' option of UNIVARIATE will be used for variables with a field width of 4 or less; and that 90-series codes will not be converted to missing.

AN EXAMPLE

The example that follows illustrates use of SASGEN. The example is divided into four parts: (1) the codebook, (2) the parameter cards, (3) the diagnostic and informational messages printed by SASGEN, and (4) the output SAS program.

Figure 2 illustrates a typical codebook one might encounter containing variable names, field widths, and variable labels. Character variables are indicated by the presence of the word 'ALPH' in columns 25-28. Note that some of the variable names are longer than 8 characters, and that some contain periods, which are not permitted in SAS. As will be seen, the SASGEN program takes into account both of these situations.

Figure 3 contains a list of the parameter cards used to describe the codebook in Figure 2 and to specify details of the analysis to be performed. The parameter cards indicate the location of the variable name, field width, and variable label in the codebook, and also specify that character variables are denoted by the presence of the word 'ALPH' in columns 25 through 28. The last 4 parameter cards dictate that: (1) 90-series codes for numeric variables are to be converted to missing values; (2) variable names containing more than 8 characters will have the first 8 characters used (i.e., they will be read 'forward'); (3) if any duplicate variable names are encountered, they will be changed; and (4) the first four variables in the codebook (which contain identification information) will be ignored in the processing.

Figure 4 contains the diagnostic and informational output provided by SASGEN. The first portion describes the codebook, in Figure 2, and to specify details of the analysis to be performed. The parameter cards indicate the location of the variable name, field width, and variable label in the codebook, and also specify that character variables are denoted by the presence of the word 'ALPH' in columns 25 through 28. The last 4 parameter cards dictate that: (1) 90-series codes for numeric variables are to be converted to missing values; (2) variable names containing more than 8 characters will have the first 8 characters used (i.e., they will be read 'forward'); (3) if any duplicate variable names are encountered, they will be changed; and (4) the first four variables in the codebook (which contain identification information) will be ignored in the processing.

The second part of the diagnostic and informational output ('MISCELLANEOUS PROCESSING INFORMATION') reports on actions taken by the program. The variable name change described results from checks made by the program for duplicate variable names (as requested by the 'DUPCHECK' parameter). Because the first 8 characters of the variable name are used, the variables Q2.REASON! and Q2.REASON2 both become Q2.REASON. In order to eliminate the duplicate character, the first 2 characters of the second of the variables are dropped and replaced with '01'. Note also that the period in the variable names have
been changed to underscores, in order to produce legitimate SAS variables. This is done for all "illegal" characters encountered.

The "Miscellaneous Processing Information" also indicates that the variable "STATE" is not included in the PROC FREQ statement generated by the program. This is in keeping with the default specification that character variables with a length greater than 4 bytes be dropped.

Figure 5 contains the SAS program created by SASGEN. Several features should be noted:

(1) The first 4 codebook variables are not included in the INPUT statement. 20 bytes are skipped before AGE is read in in order to account for this.

(2) A LENGTH statement is generated, based on the field width associated with each variable.

(3) 90-series codes have been recoded to missing. This is done by setting up arrays of numeric variables based on field width.

(4) PROC UNIVARIATE is used with the 'FREQ' option for numeric variables having a field width of 4 or less. The 'FREQ' option is not used for those variables with a width of 5 or more (Q2_REASON, Q2_REASON).

(5) PROC FREQ is used for character variables of width 3 or less (SEX). As noted earlier, the variable STATE is not included because it has a length of 10.

**SOME EXTENSIONS**

Because of space limitations, the example presented here has been a relatively simple one. For example, one capability of SASGEN that has not been
illustrated its potential for processing files with several different record types. If the codebook contains the record type (or segment number) associated with a variable, SASGEN can build a conditional INPUT statement and output records of each type to separate SAS datasets. UNIVARIATE and FREQ are then run separately on each dataset.

Although SASGEN was designed to write SAS code to perform specific types of analyses, the output from the program may be adapted to fit the user's needs. If, for example, it is desired to only build a SAS dataset and omit analysis, the appropriate statements may be deleted from the output file. Similarly, if other procedures are to be used, the appropriate statements may be substituted for the PROC UNIVARIATE and PROC FREQ statements.

**CONCLUSION**

SASGEN has been proven to be a great time-saver to the experienced SAS user as well as to the novice. In addition to providing for the routine processing of files with a very large number of variables, the ability of SASGEN to produce syntactically correct code enables jobs to be run with a minimum of debugging. In an environment where programmer time is as costly as machine time, the benefits have been extensive.
**FIGURE 5**

*generated SAS program*

```
DATA; INFILE IN;
INPUT +20
AGE 2.
SEX $ 1.
RACE 2.
STATE $ 10.
Q1_1 2.
Q1_2 2.
Q2_REASO 5.
Q2_REA01 5.
Q3 2.;

* LENGTH DECLARATIONS;
LENGTH DEFAULT=4
AGE 2 RACE 2 Q1_1 2 Q1_2 2 Q3 2;

* VARIABLE LABELS;
LABEL AGE = "RESPONDENT'S AGE (YEARS)"
SEX = "RESPONDENT'S SEX (M OR F)"
RACE = "RESPONDENT'S RACE"
STATE = "STATE IN WHICH RESPONDENT LIVES"
Q1_1 = "IS RESPONDENT SATISFIED WITH SERVICE PER"
Q1_2 = "WOULD RESPONDENT RETURN"
Q2_REASO = "1ST REASON FOR DISSATISFACTION"
Q2_REA01 = "2ND REASON FOR DISSATISFACTION"
Q3 = "LAST TIME SERVICE OBTAINED";

* RECODE 90-SERIES TO MISSING;
ARRAY TWO RACE AGE Q1_1 Q1_2 Q3 ;
DO OVER TWO;
  IF TWO=91 THEN TWO=.A;
  IF TWO=92 THEN TWO= ..B;
  IF TWO=93 THEN TWO= ..C;
  IF TWO=94 THEN TWO= ..D;
  IF TWO=95 THEN TWO= ..E;
  IF TWO=96 THEN TWO= ..F;
  IF TWO=97 THEN TWO= ..G;
  IF TWO=98 THEN TWO= ..H;
  IF TWO=99 THEN TWO= ..I;
END;

ARRAY FIVE Q2_REASO Q2_REA01 ;
DO OVER FIVE;
  IF FIVE=99991 THEN FIVE= ..A;
  IF FIVE=99992 THEN FIVE= ..B;
  IF FIVE=99993 THEN FIVE= ..C;
  IF FIVE=99994 THEN FIVE= ..D;
  IF FIVE=99995 THEN FIVE= ..E;
  IF FIVE=99996 THEN FIVE= ..F;
  IF FIVE=99997 THEN FIVE= ..G;
  IF FIVE=99998 THEN FIVE= ..H;
  IF FIVE=99999 THEN FIVE= ..I;
END;

PROC UNIVARIATE FREQ;
VAR AGE RACE Q1_1 Q1_2 Q3 ;
PROC UNIVARIATE;
VAR Q2_REASO Q2_REA01 ;
PROC FREQ; TABLES SEX ;
```