INTRODUCTION

As quick and efficient as programmers can become at producing SAS systems there are some problems that come up once the system is complete. Programmers can quickly create reports for users, however a lot of time must be invested in teaching the user enough SAS to create their own reports. This can really waste a lot of a programmer's time generating reports.

Many times what is really needed is a way to inquire against the SAS file. For example, if you want to know how many records meet a certain criteria there is no easy way to list those without creating a report. If you use FSEDII then you must press PF5/17 to see each individual record that meets the requirements. You do not have a list of the records that succeed.

There is also a limitation with timeliness. If the user needs some information and cannot write the report himself then he is at the mercy of the programmer's time constraints.

By including the SUGINQ macro in your SAS systems you allow the user to do inquiries against the system. This will have several beneficial side effects. The user will feel much more independent and will feel in control of his own SAS system. If a user can do data entry in SAS then he can use SUGINQ. Training will take only a couple of minutes. SUGINQ can be easily customized to each individual system, if desired. Proof reading of data can be very simple using this tool. SUGINQ is not intended to replace customized reports but is merely a tool to enhance system usability.

BACKGROUND

SUGINQ started out as a program for one system to be able to select drawings based on numerous selection criteria that would change from one request to another. When looking at what was required to do this type of processing a lengthy program was written specifically for this system. After showing the program to other programmers I soon found out how useful this would be to any SAS system. I proceeded to rewrite the original program for generic applications and came up with a macro program that we call SUGINQ.

EXAMPLES

The following two screens show what is seen by the user to generate a report using SUGINQ.

Description Here the user entered 11 in District, an X in Danger Pole, and an 86 in Year Inspected. Requesting a list of all danger poles for district 11 during inspection year 1986. Press PF5/21 to progress to observation 2.

Description Here the user enters a 'P' or 'Q' in each field he wants to appear on the report depending on whether the field is character or numeric.
HOW SUGINQ WORKS

To run the macro SUGINQ you must give it two parameters. The first is the SAS dataset name of the file you want to report against. The second is the screen name you want to display for the select and print criteria.

The first thing SUGINQ does is a PROC CONTENTS on the dataset that it was passed, with the options of Noprint, NOSOURCE and output to a SAS dataset.

SUGINQ then creates a macro variable of NAMES_n_, setting it equal to the variable names read in from the Proc Contents output. What SUGINQ ends up with are &NAMES1-???, where ??? is the total number of variables on the dataset. A new macro &LAST holds the last observation number.

SUGINQ then creates an empty file having the same variables as the dataset you passed it.

It then does a PROC FSEDIT on the dataset using the screen name that you specified. This is where the user inputs the selection criteria on observation 1 and print criteria on observation 2.

SUGINQ then reads the first observation from the temporary file which now has the select criteria in it. A new variable GOOD equal to 'OK' is added.

It then creates a new macro variable &VALUE1-??? for each of the variables. It does this by running a DO loop for the number of variables on the file. These macro variables hold the values of the selection criteria.

It then verifies that at least one variable was entered with selection criteria. The macro variable CHECK is used to designate that at least one select value is entered. If no selection criteria was entered for a dataset variable the NAME macro variable is blanked out. This leaves NAME macro variables from 1-???(the number of variables on the dataset) with either a dataset variable in it or blank.

BEFORE:

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;NAME1 = 'CONTRACT'</td>
<td>&amp;VALUE1 = 'X'</td>
</tr>
<tr>
<td>&amp;NAME2 = 'DANGER'</td>
<td>&amp;VALUE2 = 'P'</td>
</tr>
<tr>
<td>&amp;NAME3 = 'ENGDATE'</td>
<td>&amp;VALUE3 = '0'</td>
</tr>
</tbody>
</table>

AFTER:

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;NAME1 = 'DANGER'</td>
<td>&amp;VALUE1 = 'X'</td>
</tr>
<tr>
<td>&amp;NAME2 = 'DANGER'</td>
<td>&amp;VALUE2 = 'P'</td>
</tr>
<tr>
<td>&amp;NAME3 = 'ENGDATE'</td>
<td></td>
</tr>
</tbody>
</table>

SUGINQ then sorts the selection criteria by those variables that have selection criteria input. It does the same for the dataset passed to SUGINQ. The DO from 1 to &LAST will generate a list of all the macro variables NAMES1-???. PROC SORT will ignore those that are empty.

BEFORE:

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;NAME1 = 'CONTRACT'</td>
<td>&amp;PRINT1 = 'P'</td>
</tr>
<tr>
<td>&amp;NAME2 = 'DANGER'</td>
<td>&amp;PRINT2 = 'P'</td>
</tr>
<tr>
<td>&amp;NAME3 = 'ENGDATE'</td>
<td>&amp;PRINT3 = 'P'</td>
</tr>
</tbody>
</table>

AFTER:

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;NAME1 = 'DANGER'</td>
<td>&amp;PRINT1 = 'P'</td>
</tr>
<tr>
<td>&amp;NAME2 = 'DANGER'</td>
<td>&amp;PRINT2 = 'P'</td>
</tr>
<tr>
<td>&amp;NAME3 = 'ENGDATE'</td>
<td></td>
</tr>
</tbody>
</table>

The next process is to print dataset GOOD with the variables selected to appear on the report. This is done in a similar manner to the select variables. A PRINT macro variable is created for each variable and checked to see if a 'P' for character or a 'O' for numeric variables was entered. If neither was entered the PNAME macro variable is blanked out. This leaves the PNAME macro variables with either the dataset variable name in it or blank.

PROC SORT DATA = TEMP2;
BY
&NAMES1 =
&NAMES2 =
&NAMES3 = 'DANGER';
PROC SORT DATA = &DATA;
BY
&NAMES1 =
&NAMES2 =
&NAMES3 = 'DANGER';

Now both data sets are sorted by the selection criteria variables.

A merge of these two datasets is done using a BY statement with the same select criteria variables. The output is a dataset of all the observations. Those matching the selection criteria will have GOOD = 'OK'.

If &CHECK = OK then selection criteria was entered and only observations with GOOD = 'OK' are output to dataset GOOD, otherwise all records will be selected.

SUGINQ then creates a macro variable of PNAME_n_, setting it equal to the variable names read in from the PROC CONTENTS output. What SUGINQ ends up with is &PNAME1-???, where ??? is the total number of variables on the dataset.

&PNAME1 = 'CONTRACT'
&PNAME2 = 'DANGER'
&PNAME3 = 'ENGDATE'

The dataset is then printed using PROC PRINT and only the variables that had a 'P' entered show up in the VAR statement. If no variables are chosen then all of the variables are printed. The variables are printed in alphabetical order.
SUGINQ MACRO

%MACRO SUGINQ(DATA, SCREEN);
PROC CONTENTS DATA=DATA NOPRINT
   MOSOURCE OUT=MACRO;RUN;
DATA _NULL_; //
SET MACRO END=EOF;
CALL SYMPUT('NAME'|LEFT(_N_),NAME); 
IF EOF THEN 
   CALL SYMPUT('LAST',_N_); 
RUN;
DATA TEMP; 
SET &DATA; 
DELETE; 
RUN;
PROC FSEDIT DATA=TEMP SCREEN=&SCREEN 
   NR=24 ADD; 
RUN;
DATA TEMP; 
SET TEMP; 
LENGTH GOOD $2; 
IF _N_ = 1 THEN DO; 
   GOOD = 'OK'; 
   %DO Z=! %TO &LAST; 
      CALL SYMPUT('VALUE&Z',&&NAMES&Z); 
   %END; 
RUN;
%LET CHECK = ; 
%DO Z=1 %TO &LAST; 
   %IF &&VALUE&Z , = %THEN %00; 
      %LET CHECK = OK; 
   %ELSE %00; 
      %LET NAMES&Z = ; 
   %END; 
%ENO;
%LET CHECK = ; 
%DO Z=! %TO &LAST; 
   %IF &&NAME&Z = P OR &&NAME&Z = 0 
      %THEN %00; 
      %LET CHECK = OK; 
   %ELSE %00; 
      %LET NAME&Z = ; 
   %END; 
%ENO;
TITLE1 'METROPOLITAN EDISON COMPANY'; 
TITLE2 'POLE REPLACEMENT SYSTEM'; 
TITLE3 'SELECTION REPORT'; 
%IF &CHECK = OK %THEN %00; 
PROC PRINT DATA=GOOD;2 GRI0NO; 
   VAR %DO Z=1 %TO &LAST; 
      &&NAME&Z 
   %END; 
%STR(,); 
RUN;
%END; 
%EN0; 
%MEN0 SUGINQ;
MODIFICATIONS TO SUGINQ

The above was an example of the macro SUGINQ without any modifications. The macro was passed the SAS dataset name and the screen. This is alright for most systems, but some might require more than the basic SUGINQ. The following is an example of changes that can be easily made to SUGINQ to make it even more responsive to the user's needs.

Because no actual variable name is used in the processing, this macro can be used for any SAS dataset to create user generated inquiries. The reports are not fancy with nice labels, but will usually meet ninety percent of user requests and free programmers for more important and challenging tasks.

Other modifications possible:

• Creation of new variables - Add new variables to the data set for selecting or reporting purposes. An example would be a substring of a variable to limit the length.

• Ability to enter Title of Report - Request for input of a title.

• Labels on output - You can add the split = option on your print to use labels for column headings. An easy way to do this is to use proc datasets on the data set and then the variables option to enter labels for the variables.

• Different Sort criteria - Request user for sort criteria.

• Other print options - An example of this is to sum on a particular variable.

• Other Proc statements - Any proc statement can be included in SUGINQ to process the selected records.

CONCLUSION

SUGINQ has not left SAS programmers with no reporting to do, but has relieved some of the pressure that comes with inquiry type reporting. These types of reports almost always need to be run yesterday. SUGINQ also allows for the user to feel more self-reliant and more supportive of the system. Programming is still required for reports that go to top management or those that are required every month. SUGINQ results in the majority of requests now being handled where they belong, by the Users of the system.

FOR MORE INFORMATION

Andy Schlegel or Barbara Lee
Metropolitan Edison Company
P.O. Box 16001
Reading, PA 19640

SAS is a registered trademark of SAS Institute Inc., Cary, NC, USA.