TECHNIQUES FOR USING THE SAS® SYSTEM IN AN ON-LINE PRODUCTION ENVIRONMENT

Barbara Kennedy, SAS Institute Inc.

Yes, you can operate a company using the SAS® System.

The Management Information Systems Department at SAS Institute provides a variety of programming services to every department and person at the Institute. There are over 22 departments who regularly use applications developed by MIS. All of these systems were written using SAS software. Included among these departments are Administration, Finance, Education, Sales, Legal, Marketing, and Operations.

There are approximately 150 employees out of 500 actually logged on and executing these SAS programs on a daily basis. These systems include book orders, inventories, work schedules, documentation libraries, software sales, payroll, fixed asset accounting, personnel applicants, and everything else a company needs to operate on a day-to-day basis.

The MIS Department has established guidelines or goals for system design to include:

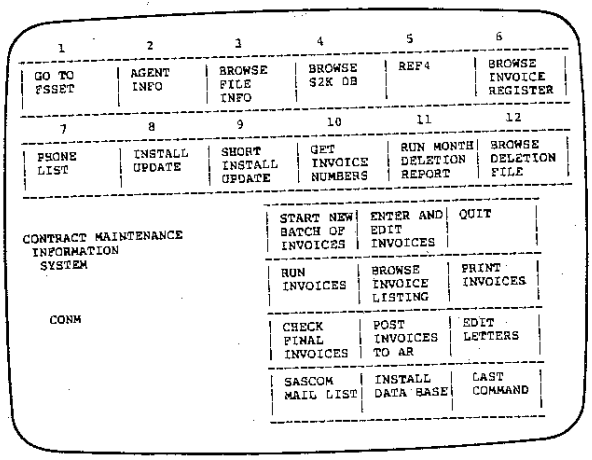
- consistency of design
- using a modular approach to programming (which provides for)
- ease of maintenance
- flexibility (and)
- user control.

This is accomplished by using menu-driven systems written with base SAS software, making use of its macro facility. We have successfully used this design under both TSO and CMS.

These systems can be as simple as a screen representing all the reports available to the user during end-of-the-month processing, to the more complex invoicing systems. The menu screen provides the link between the users' tasks and the programming code. Users appreciate the flexibility and responsibility given them when using these systems. They can, based on priorities set by them or their supervisors, determine what tasks will be performed and when. They can manage the processing of their work, based on the status of related activities; they maintain control over their job. If they aren't able to close the books on the normal schedule, they have the power to delay running monthly reports. If no transactions for updating a file were entered on that day, they don't have to submit a job to update the file. Things don't simply happen to these users and they appreciate it.

We can look at an example, the Contract Maintenance system, to see just how this fits together. This group of users is responsible for invoicing new products, updating the installation data base, and helping customers with billing questions. In order to do these tasks, the user must have access to a number of master data sets, as well as their working files and reports. All of these resources are available through one command that brings in their screen. Then at the touch of a key, or entry of a macro name, they are able to perform a specific function.

CONTRACT MAINTENANCE MENU SCREEN



Behind the user's screen is modularized code in the form of macros, each one invoking code necessary to provide the user the ability to perform their desired task. These macros may contain code that submits a job for later execution, brings in SAS code that runs interactively, or executes one of the many SAS System products.

```
FILE: CONM     SAS      *

*-------------------------------------------------------------------------*
*                                                                         *
* PROGRAM NAME: CONM                                                      *
*                                                                         *
* SUPPORT: MIS - FIN                                                      *
*                                                                         *
* SYSTEM:  CONTRACT MAINTENANCE                                           *
*                                                                         *
* REQUEST #:                                                              *
*                                                                         *
* PURPOSE: BASIC SYSTEM FOR CONTRACT MAINTENANCE FOR UPDATING IDB         *
*          AND INVOICING FOR IPPS                                         *
*                                                                         *
* NOTES:                                                                  *
*                                                                         *
*-------------------------------------------------------------------------*
*                                                                         *
* MODIFICATIONS                                                           *
*                                                                         *
* REQUEST #     DATE     PROGRAMMER     DESCRIPTION                      *
* ---------     ----     ----------     -----------                      *
*                                                                         *
*-------------------------------------------------------------------------*;
OPTIONS SORTPGM=SAS OQUOTE MPRINT SYMBOLGEN;
%INCLUDE CBINARY;
%INCLUDE FBINARY;
%INCLUDE GLOBID;

*PF1;
*USED FOR CMS SESSION TO SWITCH BETWEEN IDS;

*PF2;
MACRO _AGINFO
      CMS GETSAS AGINFO;
      RUN;
%

*PF3;
MACRO _BRCFILE
      CMS GETSAS BRCFILE;
      RUN;
%
```

```
*PF4;
MACRO _S2KIDB;
      CMS GETSAS S2KIDB;
      RUN;
%

*PF5;
%MACRO RREF4;
      %INCLUDE REF4;
%MEND RREF4;

*PF6;
MACRO _BRINV
      CMS GETSAS INVOICES;
      RUN;
%

*PF7;
MACRO _PHONE
      CMS GETSAS PHONE;
%

*PF8;
MACRO _INSTUP
      CMS GETSAS INSTUP;
%

*PF9;
%MACRO SINSTUP;
      %INCLUDE SINSTUP;
%MEND SINSTUP;

*PF10;
%MACRO GETINV;
      %INCLUDE GETINV;
%MEND GETINV;

*PF11;
MACRO _DELREP
      CMS GETSAS DELREP;
      RUN;
%

*PF12;
MACRO _DELIDB
      CMS GETSAS DELIDB;
      RUN;
%

*PF13;
MACRO _NEW
      CMS GETSAS NEWI;
      RUN;
%

*PF14;
MACRO _EINV
      CMS GETSAS EINV;
%

*PF15;

*PF16;
%MACRO RINV;
      %INCLUDE CONINV;
%MEND RINV;

*PF17;
MACRO _BINV
      CMS BROWSE INVOICES LISTING;
      RUN;
      _MSG
      RUN;
%

*PF18;
MACRO _PINV
      *CMS SP PRINT FORM 4860;
      CMS SP PRINT FORM 4861;
      CMS PRINT INVOICES LISTING;
      RUN;
      CMS SP PRINT FORM STANDARD;
      _MSG
      RUN;
%

*PF19;
MACRO _AREC
      CMS EXEC GETSASDB AREC&LOGID WRITE;
      RUN;
      PROC FSEDIT DATA=AREC&LOGID..INVOICES SCREEN=AREC&LOGID..ARSCR2
         OPT=1;
      RUN;
      DATA A;
         SET AREC&LOGID..INVOICES;
         IF INVOICE=' ' & PREFIX=' ' & INVDATE=. & INVAMT=.
            THEN DELETE;
      RUN;
      DATA AREC&LOGID..INVOICES;
         SET A;
      RUN;
      CMS EXEC RETSASDB AREC&LOGID;
      RUN;
      _MSG
      RUN;
%
```

```
*PF20;
MACRO _SEND
      CMS BATCH SUBMIT SEND JOB;
      _MSG
      RUN;
%

*PF21;
%MACRO _LETTER;
      %INCLUDE LETTER;
      RUN;
%MEND _LETTER;

*PF22;
MACRO _SASCOM
      CMS GETSAS SASCOM;
%

*PF23;
%MACRO INSTAL;
      %INCLUDE INSTALLC;
%MEND INSTAL;

*PF24;

MACRO _XXXX
      _MSG
      RUN;
%

MACRO _MSG
      CMS EXECSERV CLEAR;
      DATA _NULL_;
        FILE PRINT NOTITLES;
        PUT
'      1          2          3          4          5          6          '/
' ------------------------------------------------------------------------'/
' | GO TO    | AGENT    | BROWSE   | BROWSE   | REF4     | BROWSE   |'/
' | FSSET    | INFO     | FILE     | S2K DB   |          | INVOICE  |'/
' |          |          | INFO     |          |          | REGISTER |'/
' ------------------------------------------------------------------------'/
'      7          8          9          10         11         12         '/
' ------------------------------------------------------------------------'/
' | PHONE    | INSTALL  | SHORT    | GET      | RUN MONTH| BROWSE   |'/
' | LIST     | UPDATE   | INSTALL  | INVOICE  | DELETION | DELETION |'/
' |          |          | UPDATE   | NUMBERS  | REPORT   | FILE     |'/
' ------------------------------------------------------------------------'/
'                                 ------------------------------------   '/
'                                 | START NEW| ENTER AND| QUIT     |   '/
' CONTRACT MAINTENANCE            | BATCH OF | EDIT     |          |   '/
'  INFORMATION                    | INVOICES | INVOICES |          |   '/
'   SYSTEM                        ------------------------------------   '/
'                                 | RUN      | BROWSE   | PRINT    |   '/
'                                 | INVOICES | INVOICE  | INVOICES |   '/
'                                 |          | LISTING  |          |   '/
'     CONM                        ------------------------------------   '/
'                                 | CHECK    | POST     | EDIT     |   '/
'                                 | FINAL    | INVOICES | LETTERS  |   '/
'                                 | INVOICES | TO AR    |          |   '/
'                                 ------------------------------------   '/
'                                 | SASCOM   | INSTALL  | LAST     |   '/
'                                 | MAIL LIST| DATA BASE| COMMAND  |   '/
'                                 |          |          |          |   '/
'                                 ------------------------------------   ';

      RUN;
%

_MSG
RUN;
%INCLUDE *;
```

One of the more interesting and most used macros is the one we have labeled _MSG. This macro contains the code that generates the screen to tell the user their choices of tasks available under their system. There can be multiple screens for a single system called through one of the command keys. All of the other macros end with an additional call to the screen macro, _MSG.

Everyone's favorate task is the one to quit. This is simply a defined PF key or a command to end execution of the SAS session and return to the operating system.

Using the macros with the %INCLUDE command allows the same code from a single source to be used in different systems. Independent systems may share some of the same programs when their tasks overlap. For example, we have three systems that work with invoicing at various levels and stages. All three of these groups

use the same program code from a single source for the same task, even though they have other activities with code not available between groups. Using the same code from a single source in different systems is a simple thing to do and further reduces maintenance time.

Structuring systems with macros in this manner provides to the programming staff a single reference point for a system, allowing for ease of maintenance and enhancements, even for the programmer unfamiliar with the specific system, but famliar with their common design. This design provides to the MIS staff an index to the complete system, making the often complex relationships within an application more visible and more understandable.

Common situations arise in production environments. Some of these production considerations are updating of master data sets (especially when there are multiple users), searching large data files, and tracking changes to these master data sets. We successfully maintain data sets through the use of transaction files. Users with the authority to update these master files are given their own transaction data sets that consist of either observations retrieved from the master data sets or empty data sets duplicating the variables that can be updated.

Retrieved data sets is our terminology for selecting from large master files specific observations and putting them in the user's own data set, thus giving the user a subset of the master file in which to edit and update information.

An empty transaction file is a duplicate of the variables on the master data set with the values not filled in. The user enters on these screens the matching or identifying values along with any value that needs changing on the master file.

Prior to the actual update of the master data set, a variety of editing and reporting can be generated using the information on these transactions. The extent of editing and reporting is based on the user's requirements and is easily adjusted with changing situations. At a later point in time, either on an automated schedule or under the user's control, these transaction files are used to merge with or update the master data files. These updates are generally scheduled for late night when the computers have a lighter load and users no longer need access to the files.

Varying with the need and importance of changes to a particular data set, the transaction files can be saved and used as audit trails. Tracking information is enhanced by adding information to these transactions, such as the current date, the userid, and the transaction type (add, delete, or change). Depending on the need, these transactions are accumulated over time, from a few days to forever. When these files are used, they can supply us with a complete history of data modifications. The transaction files from the empty data set provide us with a permanent record of both the key fields and the specific changes that were made. On the retrieved transaction, we know what the observation was changed to look like. On both, we know by whom and when the changes were made. These saved update files can either be stored in case of future need or used as an ongoing data set to provide activity information for a department. In some applications, we have created new data sets composed of observations slated for deletion. One of these files is used as a reference for equipment no longer on active inventory.

Searching large files for specific records based on an id or key can be time-consuming. We have two methods of selecting records that work well, the merge statement and binary searches. These selection methods may simply bring records into a work file for temporary browsing or into a permanent file that can then be edited as a transaction file. Some of our larger data sets have multiple records that are related to each other or may be needed as references when working with them. By retrieving these related observations, the user can build a unique file of references to make use of the files and time more efficient and organized. It is much easier to work if the ten or twenty observations specific to your needs are available without other unrelated data mixed in between.

The merge method is the simpler method to understand and code. It uses a file to enter id or key values then merges with the master, keeping those observations that match.

```
FILE: INVUPDT2 SAS      A

*-------------------------------------------------------------------------*
*                                                                         *
* PROGRAM NAME: INVUPDT2                                                  *
*                                                                         *
* SUPPORT: MIS - FIN                                                      *
*                                                                         *
* SYSTEM:  CASH RECEIPTS                                                  *
*                                                                         *
* REQUEST #:                                                              *
*                                                                         *
* PURPOSE: THIS PROGRAM PREPARES A DATA SET TO BE USED FOR UPDATING       *
*          THE MASTER INVOICE FILE --                                     *
*          STEP ONE: THE USER ENTERS THE INVOICE NUMBER AND PREFIX        *
*          STEP TWO: THE PROGRAM BRINGS IN RECORDS FROM THE INVOICE       *
*          FILE CORRESPONDING TO THE RECORDS IN STEP ONE                  *
*          STEP THREE: THE USER MAKES CORRECTIONS ON THIS FILE            *
*          AFTER SUBMISSION OF A NIGHT JOB THESE CHANGES ARE REFLECTED    *
*          ON THE MASTER INVOICE FILE                                     *
**                                                                        *
* NOTES:                                                                  *
*                                                                         *
*-------------------------------------------------------------------------*
*                                                                         *
* MODIFICATIONS                                                           *
*                                                                         *
* REQUEST #     DATE     PROGRAMMER     DESCRIPTION                      *
* _________     ____     __________     ___________                      *
*                                                                         *
*-------------------------------------------------------------------------;

PROC FSEDIT DATA=ARECADM7.INVTRAN SCREEN=AR.IVUP1SCR OPT=1;
PROC SORT DATA=ARECADM7.INVTRAN;
   BY PREFIX INVOICE CONT;
DATA INVOICES;
   MERGE ARECADM7.INVTRAN(IN=IN2) AR.INVOICES(IN=IN1);
   BY PREFIX INVOICE ;
   IF IN2;
DATA INVOICES;
   SET ARECADM7.INVOICES INVOICES;
PROC FSEDIT DATA=INVOICES SCREEN=AR.INVUPSCR OPT=1;
   FORMAT ARACCT1-ARACCT15 AACCOUNT23.;
   MISSING _ M;
RUN;
DATA ARECADM7.INVOICES;
   SET INVOICES;
   IF INVOICE=' ' & COMPANY=' ' & INVAMT=. & CXNO1=' '  THEN DELETE;
RUN;
DATA ARECADM7.INVTRAN;
   SET ARECADM7.INVTRAN(OBS=0);
_MSG;
RUN;
```

The other method we use on a regular basis is a form of binary search that searches the master data set and pulls off the matching records to create either a work (temporary) file or a permanent file.

```
FILE: BINARY   SAS      A


%MACRO BINARY;
  * DATA SET TEMP CREATED IN THE MACRO THAT CALLS THIS MACRO;
  * AND CONTAINS THE DATA OF WHO THE SEARCH IS FOR;
DATA _NULL_;
   SET TEMP;
  * DEFINE SYMBOLIC VARIABLES FOR THE CHOICES OF THE USER;
  * VAR1=WHICH ID (KEY) WILL BE USED;
 * CREATE SYMBOLIC VARIABLE THE NAME OF KEY;
 CALL SYMPUT('VAR',VAR1);
DATA CURR;
IF _N_=1
 THEN SET TEMP;
  /*SET DEFAULT VALUES INCASE INCORRECT VALUES FOR ID CHOICE IS SENT*/
 %LET INVAR=REF4;
 %LET DATASET=COMBKUP;
 %LET DATAOUT=INSTALL.COMBKUP;
 %LET INDEX1=1;
 %LET INDEX2=1;
  /* SET  VALUES IF REF4 IS THE ID (KEY) */
 %IF %UPCASE(&VAR)=R
  %THEN %DO;
    %LET INVAR=REF4;
    %LET DATASET=COMBKUP;
    %LET DATAOUT=CURR;
    %LET INDEX1=2;
    %LET INDEX2=40;
  %END;
   /*SET  VALUES IF SITE IS THE ID (KEY) */
  %ELSE %IF %UPCASE(&VAR)=S
   %THEN %DO;
    %LET INVAR=SITE;
    %LET DATASET=COMBINE;
    %LET DATAOUT=CURR;
    %LET INDEX1=2;
    %LET INDEX2=40;
   %END;
 ARRAY VAR (X) $ 9 VAR1-VAR40;
 /* CONVERT INPUT DATA TO UPPERCASE VALUES */
 DO OVER VAR ;
  VAR=UPCASE(VAR);
 END;

 DO X=&INDEX1 TO &INDEX2;
  IF X=1 THEN STOP;
  IF VAR=' ' THEN STOP;
  HIGH=N; *SET HIGH VALUE TO NUMBER OF OBSERVATIONS;
  LOW=1;
  MID=CEIL(HIGH/2); *CALCULATE MIDPOINT BETWEEN HIGH AND LOW;
  DO M=1 TO N;
   SET INSTALL.&DATASET POINT=MID NOBS=N;
     *COMPARE THE VARIABLE FROM THE DATA SET BEING READ;
     *INVAR IS THE VARIABLE KEYED IN BY THE USER;
     *RESET HIGH AND LOW DEPENDING ON VALUE OF THE KEY;
     IF &INVAR<VAR THEN LOW=MID;
     ELSE IF &INVAR>VAR THEN HIGH=MID;
     ELSE DO; *MATCH FOUND;
       SAVEMID=MID;
       OUTPUT CURR;
       GO TO ENDLOOP;
     END;
     MID=CEIL((HIGH-LOW)/2+LOW); *CALCULATE MIDPOINTS;
     *CAN NO LONGER SPLIT FILE FOR SEARCHING AND MATCH NOT YET FOUND;
     IF MID=LOW | MID=HIGH
      THEN DO;
       ND=0; *SET FLAG;
       MID=1; *CHECK TO SEE IF EQUAL TO FIRST OBSERVATION;
       SET INSTALL.&DATASET POINT=MID;
       IF &INVAR=VAR
        THEN DO; *MATCH FOUND;
         ND=1; *SET FLAG;
         OUTPUT CURR;
        END;
        MID=N; *CHECK TO SEE IF EQUAL TO LAST OBSERVATION;
        SET INSTALL.&DATASET POINT=MID NOBS=N;
        IF &INVAR=VAR
         THEN DO; *MATCH FOUND;
          ND=1; *SET FLAG;
          OUTPUT CURR;
         END;
        IF ND=0
         THEN DO; *STILL NO MATCH;
          *NOTIFY USER NO RECORD IS FOUND;
          *BY SETTING AN OBSERVATION TO DUMMY DATA;
          COMPANY='NO MATCH FOUND FOR '|| VAR;
          NAME   ='NO MATCH FOUND FOR '|| VAR;
          SITE='XXXX';
          REF4='XXXX';
          OUTPUT CURR;
        END;
        GO TO ENDLOOP;
      END;
 END;
ENDLOOP: END;
DROP HIGH LOW SAVEMID M VAR1-VAR40 X;
STOP;
RUN;
 * DISPLAY MATCHES ON TERMINAL WITH FSEDIT;
PROC FSEDIT DATA=&DATAOUT SCREEN=INSTALL.ALLSCN OPT=1;
    MISSING M;
    FORMAT EXCEPTN $MEXCEPT.;
RUN;
%MEND BINARY;
```

This mode of creating subset data files for individual users gives them a starting point for editing, as well as a quick reference file for use in the daily processing of information.

We know from experience that these design methods work, and in combination with the easy-to-use SAS software, provide us with flexibility, ease of maintenance, and satisfied users. Presently, we are converting our current production systems to run under the newest release of the SAS system, with the core of our development being SAS/AF™ software. Because of the modular approach we have taken in system design, the transfer to our new software products is going smoothly and we will be even less operating-system dependent, and able to provide even easier, productive applications using the SAS System.