Paper 209-26

SAS® Stew Keeps Facilities Running

Michael A. Mace, SPS Software Services, Inc., Canton, OH

ABSTRACT

At first glance, this application looks like five SAS/FSEDIT® screens and several canned reports. However, when we look at the recipe, we find oodles of nifty things working together --- from SAS/ACCESS® for DB2™. SCL assisting in data entry, window-jumping, and data summing to reporting and data maintenance. So many good things, we have to come back for seconds.
Please note that while this application was written using version 6.09 on MVS, the tools and techniques are just as applicable on other platforms and with later versions of the SAS® system.

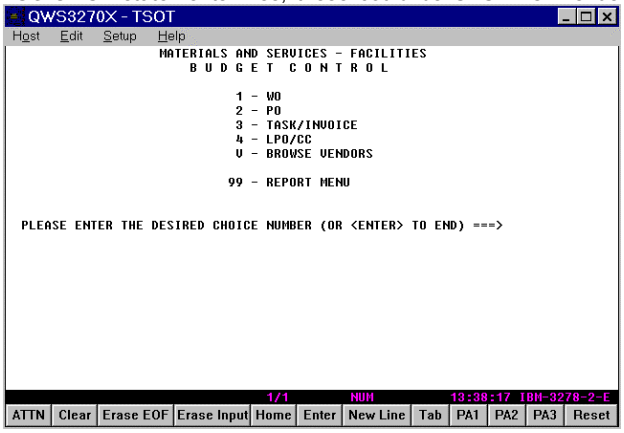
INTRODUCTION

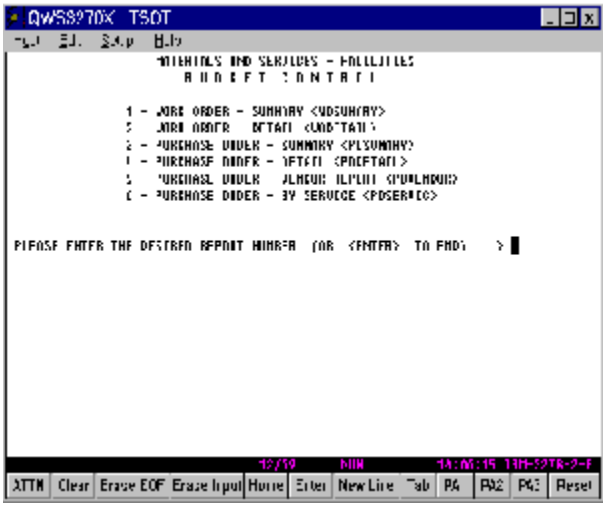
It is very possible that I should have titled this paper “A Recipe for SAS® Jambalaya” or “The Kitchen Sink of SAS® Applications” because of the variety of tools from the SAS® system that are used. If you have not guessed by now, I enjoy good eating as much as I enjoy working with the SAS® system. As you can see by a glance at the reference section, I will leave it to you to explore the detailed explanations about syntax, usage, et cetera; I want to let you sample as many appetizers as possible in the time and space allotted.

The facilities department of a certain company needed an application which they could use to track their budget and expenditures on a more-or-less real-time basis; instead of waiting until sometime into the next month for the reports from accounting. Essentially, there is a need to enter, track, and report on budgeted, committed, and invoiced dollars by work order, purchase order and task. Much of data that is entered into the accounting system originates in the department, and while it means entering that data twice (because there was no simple means to tie the systems together), the folks in facilities determined that it was worth it, if the application could automate some tasks, be user-friendly and forgiving, and reliable. I assured them that, with the SAS® system, I could create an application they would savor.

Loosen your belts and put on your bib . . .

These two screens are the front-end menus, produced with TSO/CLIST statements. Yes, these could be SAS/AF® menus.





This second screen is the report menu. For the sake of space, let me show you just one of the programs; please notice the variety of SAS® tools in just this course. Go ahead, steal the code; just send me some blackberry pie or a certificate for a restaurant where I can get some bouillabaisse.

```
LIBNAME facilitys 'msms.facilitys.PERM.SASDATA'
          SERVER= SHARE6P  ;
LIBNAME library 'msms.formats.PERM.SASDATA'
          SERVER= SHARE6P  ;

FOOTNOTE "<wodetail>" ;

CLEAR ;
%PUT################################################ ;
%PUT Please enter the desired WO SERIAL number, i.e. 12345 ;
%INPUT woserial ;

PROC  MEANS DATA= facilitys.taskinvc
                        (WHERE=(woserial="&woserial"))
                    SUM NOPRINT ;
   VAR committd invoiced ;
   CLASS potask ;
   ID woferc wo vndrname ;
   OUTPUT OUT=tskinv (RENAME= (wo=wox))  SUM= ;
RUN ;

DATA wotskinv ;
   IF _N_ = 1
   THEN SET facilitys.wo (WHERE=(woserial="&woserial")
                          KEEP= woserial wo11 descrpt1) ;
   SET tskinv (WHERE=(_TYPE_ GT 0)) ;
   CALL SYMPUT('wo', wo11) ;
   CALL SYMPUT('descrpt', descrpt1) ;
RUN ;

PROC SORT DATA= wotskinv   NODUPLICATES ;
   BY woferc potask ;
RUN ;
```

```
OPTIONS NOCENTER LINESIZE=100 ;
TITLE1 "Facilities Management: WO Detail" ;
TITLE2 "&wo" ;TITLE3 "&descrpt" ;

PROC PRINT DATA= wotskinv   NOOBS UNIFORM LABEL ;
    VAR potask vndrname committd invoiced ;
    BY woferc ;
    ID woferc ;
    SUMBY woferc ;
    FORMAT committd invoiced DOLLAR15.2 ;
    LABEL potask ='po task'
          vndrname ='vendor'
          committd ='committed'  ;
RUN ;

* ################################################ ;
*    WOSUMMARY ;

PROC MEANS DATA= faclitys.taskinvc
                    (WHERE= (woserial="&woserial")
                     KEEP= wo19 woserial committd invoiced)
                SUM NOPRINT ;
    VAR committd invoiced ;
    CLASS wo19 ;
    OUTPUT OUT= tskinv   SUM= ;
RUN ;

DATA wo (KEEP= wo wo11 wo19 budget bucket) ;
     LENGTH bucket $10 ;
     SET faclitys.wo (KEEP= wo wo11 woserial budget1-budget15
                            wo19_1-wo19_15 woferc1-woferc15
                      WHERE= (woserial="&woserial") ) ;
     ARRAY wo19x(*) wo19_1-wo19_15 ;
     ARRAY wofercx(*) woferc1-woferc15 ;
     ARRAY budgetx(*) budget1-budget15 ;

     DO i = 1 TO 15 ;
        IF wo19x(i) NE ''
        THEN DO ;
                wo19 = wo19x(i) ;
                budget = budgetx(i) ;
                IF wofercx(i) =: '3' THEN bucket = 'Capital' ;
                ELSE IF wofercx(i) =: '184'
                      THEN bucket = 'O & M' ;
                ELSE IF wofercx(i) =:'1082'
                      THEN bucket = 'Retirement';
                OUTPUT ;
             END ;
     END ;
RUN ;

PROC SQL ;
    CREATE TABLE wotskinv AS
    SELECT wo, wo11, bucket, budget, committd, invoiced
    FROM wo  LEFT JOIN  tskinv
    ON wo.wo19 = tskinv.wo19
        AND _TYPE_ GT 0
    ORDER BY wo.wo
   ;
QUIT ;
RUN ;
TITLE1 "Facilities Management: WO Detail" ;
TITLE2 "Summary for:   &wo" ;
PROC TABULATE DATA= wotskinv  FORMAT= DOLLAR15.2 ;
    CLASS wo bucket ;
    VAR budget committd invoiced ;
    KEYLABEL SUM =' '
                ALL ='Total'  ;
    LABEL committd ='committed' ;
    TABLE wo=' ' * (bucket=' ' ALL),
          (budget committd invoiced) * SUM ;
RUN ;
LIBNAME library CLEAR ;
LIBNAME faclitys CLEAR ;
```

Here is a portion of another program; just to show another reporting technique.

```
FOOTNOTE "<podetail>" ;
.
.
.
DATA _NULL_ ;
    FILE PRINT   HEADER=hdr LINESLEFT=ll ;
    SET potskinv ;
    ARRAY detail(*) detail1-detail10 ;

    IF ll LT 10 THEN PUT _PAGE_ ;
    PUT / @1 task  @5 wo  @30 descrpt
          @100 committd DOLLAR15.2
          @115 invoiced DOLLAR15.2 ;
    DO i = 1 TO 10 ;
       IF detail(i) NE '' THEN PUT @30 detail(i) ;
    END ;
RETURN ;

hdr: ;
    PUT  @55 "Facilities Management: PO Detail"
         / @55 "&po   &vndrname"
         // @1 'task' @5 '      wo'  @30 'description / details'
           @100 '    committed'  @115 '     invoiced'
           ;
RETURN ;

RUN ;
```

Now that I have shown you the dessert tray (well, don't you look to see for what you might want to save room), let's go back to the beginning.

APPETIZERS

There are several people in the department who enter data and in fact, there are several locations using this application; therefore SAS/SHARE® is used in each program to allow concurrent access to the data sets. I provide access to the format library with SAS/SHARE® also because key clients have the ability to modify the format library with a utility program that I will show you shortly.

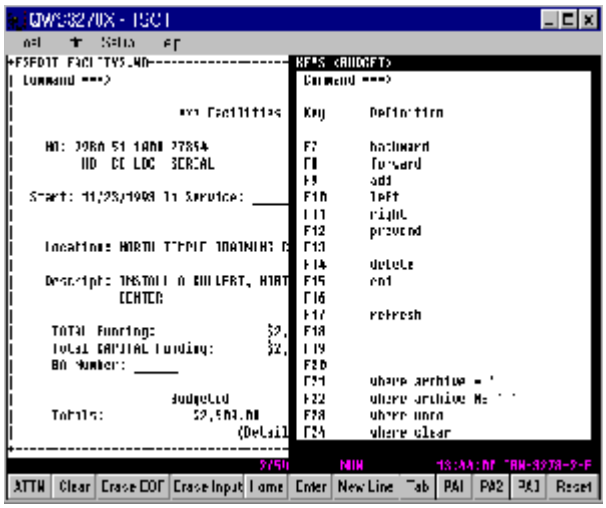
Vendor information for the entire company is kept in a DB2™ database on a separate operating system than that on which the SAS® system is run. Therefore a nightly job is executed to refresh that data into a SAS® data set for immediate, read-only, lookup access by the facilities application. SAS/ACCESS® was used to create the access and view descriptors that implement this process.
With later versions of the SAS® system, the LIBNAME statement or SQL Pass-through may have been used. Please refer to *SAS/ACCESS ® INTERFACE to DB2™: Usage and Reference, Version 6, First Edition* .

The first menu screen simply determines with which data set the user wants to work; the data set name is passed to the following program via the &SYSPARM option of the SAS® invoking statement. The libref, affspscl, for the screen library is allocated in the TSO/CLIST.

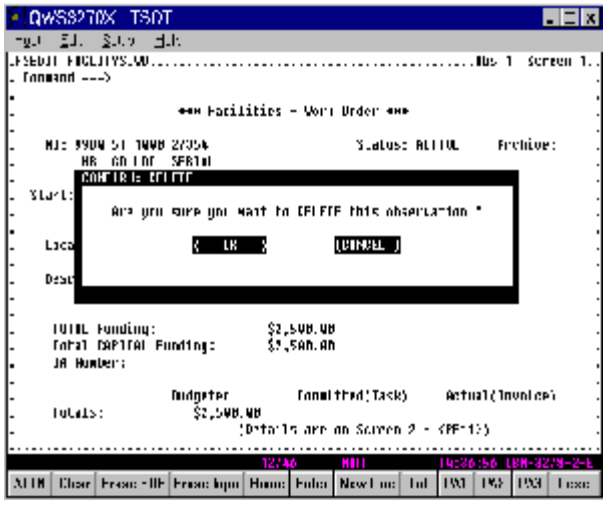
```
LIBNAME faclitys 'msms.faclitys.PERM.SASDATA'
          SERVER= SHARE6P  ;
LIBNAME library 'msms.formats.PERM.SASDATA'
          SERVER= SHARE6P  ;

%MACRO msmsedit ;
%LET sysuid = %SUBSTR(&SYSPARM,1,5) ;
%LET dsn = %SUBSTR(&SYSPARM,6) ;
```

```
%IF &dsn = VENDORS
%THEN %DO ;
                PROC FSBROWSE DATA= faclitys.&dsn
                SCREEN= affspscl.budscrns.&dsn..screen ;
                RUN ;
          %END ;
%ELSE %DO ;
                PROC FSEDIT DATA= faclitys.&dsn
                SCREEN= affspscl.budscrns.&dsn..screen ;
                RUN ;
           %END ;
%MEND msmsedit ;
%msmsedit ;
LIBNAME library CLEAR ;
LIBNAME faclitys CLEAR ;
```

Each FSEDIT screen utilizes a customized set of PFKEYS, created with PROC BUILD of the SAS/AF® software. The users can access the key definitions by typing KEYS on the command; the display would look like . . .



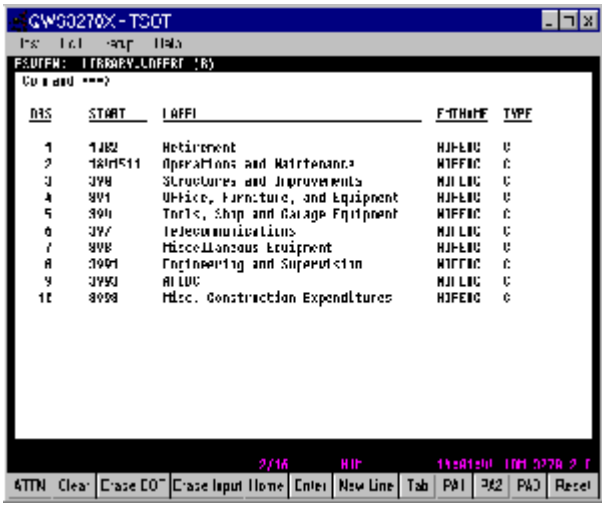
In addition to several LIST entries, PROC BUILD was also used to create a basic SAS/AF® program entry with a choice group of type ACTION, which is used to confirm that the user really wants to delete an observation. Please see *SAS/AF ®Software: Usage and Reference, Version 6, First Edition.* The pop-up window is displayed thusly . . .



The utility program for maintaining the format library is:

```
LIBNAME library 'msms.FORMATS.PERM.SASDATA'
            SERVER= SHARE6P  ;

/*  Code like this was used to initially create the SAS data sets
and format library */
DATA typefrmt ;
      SET msms.formatname ;
      TYPE = 'C' ;  -OR-  TYPE = 'N' ;
      FMTNAME = '$formatname' ;
      -OR- FMTNAME = 'formatname' ;
RUN ;
PROC FORMAT LIBRARY= library  CNTLIN= typefrmt ;
RUN ;

PROC FORMAT LIBRARY= library ;
      PICTURE  phone  OTHER='(000) 000-0000' ;
RUN ;
  */
%MACRO frmt ;
CLEAR ;
%PUT Please enter the name of the format dataset to edit ;
%INPUT frmtname ;

%IF &frmtname NE ''
%THEN %DO ;
                PROC FSVIEW DATA= library.&frmtname
                                MODIFY;
                RUN ;

                PROC FORMAT LIBRARY= library
                                CNTLIN= library.&frmtname ;
                RUN ;
           %END ;

* to get a printout of the formats ;
PROC PRINTTO FILE='msms.formats.printout'   NEW ;
RUN ;
PROC FORMAT LIBRARY= library  FMTLIB PAGE ;
RUN ;
PROC PRINTTO ;
RUN ;

%MEND frmt ;
%frmt ;
LIBNAME library CLEAR ;
```

The SAS® data sets do not have to be in the same library as the formats; I chose to put them there for convenience. When the clients execute this program, a screen similar to the following is displayed.



Main Course: SCL - Screen Control Language or SAS® Component Language

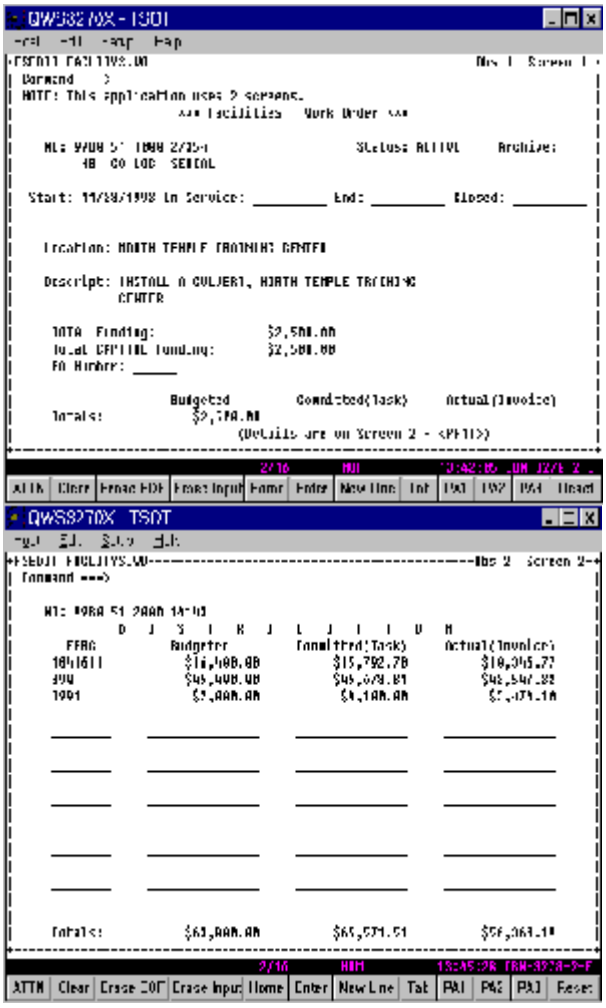
Once again, this application essentially records and reports on work order, purchase order, and related task data. As I present the screens and the SCL code for each, remember that the FSEINIT and FSETERM sections execute at the beginning and end of the FSEDIT session, the INIT and TERM sections execute for each observation, the MAIN section executes whenever the [ENTER] key is pressed, and the labeled sections execute when the respective screen variable fields are edited.

For ease of maintenance and because the screen catalog library is allocated for the clients with a disposition of share, I keep the SCL programs in a data set along with all the other programs for the application. The only code that I put behind the screen is a statement like this:

%INCLUDE “fully-qualified_data_set_name(SCLprogramname)” ;

This way I can easily view and modify the code; then when the users are out of the application I can allocate the screen catalog library with exclusive access and within one minute implement any changes. While this application does not use any SAS/AF® program screens with SCL, I have found that this technique really helps in those applications !

Work Order



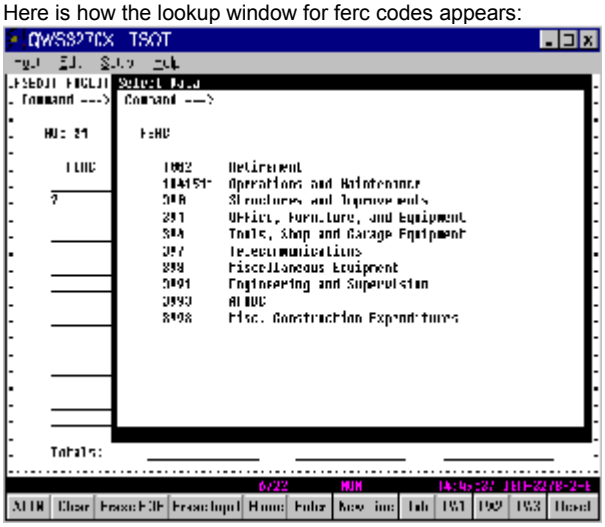
This second screen summarizes all monetary data for each work order. The Committed(Task) and Actual(Invoice) fields are protected, refreshed by a program (to be presented later) executed nightly or the user can enter the word ‘refresh’ on the command line; the code to support this is in the following SCL program.

```
/* MSMS.SASPROG.PERM.SAS(SCLWO) */

LENGTH response $ 1   /* This is for the DELETE
                         confirmation window */
          invctmp1 invctmp2  8  ;
FSEINIT:
    dsiself = OPEN('faclitys.wo','i') ;

  /* to enable screen variable labels and ensure that the MAIN
section is always executed */
    CONTROL LABEL ALWAYS ;
RETURN ;


INIT:
   ARRAY woparts(4) $8 ('wohb','woco','woloc','woserial') ;
   ARRAY wo19x(15) $19 wo19_1-wo19_15 ;
   ARRAY woferc(15) $8  woferc1-woferc15 ;
   ARRAY commit(15) 8  commit1-commit15 ;
   ARRAY invoic(15) 8  invoic1-invoic15 ;
RETURN ;


status:
  /* determines the size of the next window opened */
   CALL WREGION(10,35,15,45,'') ;

  /* if the user enters a ‘?’ or a [SPACE] in a field, display a pop-
up list, created with SAS/AF® */

   IF status IN('?',' ')
   THEN status = LISTC('status.list','','Y',1) ;
RETURN ;

archive:
   IF archive NE ''
   THEN archive ='X' ;
RETURN ;

woco:
   CALL WREGION(10,35,15,45,'') ;
   IF woco IN('?',' ')
   THEN woco = LISTC('company.list','','Y',1) ;
RETURN ;


MAIN:
   cmdword = WORD(1,'U') ;   * command line,first word ;

   IF cmdword =: 'DEL'
   THEN DO ;       /* here is the confirmation displayed earlier */
             response = 'N' ;
             CALL WREGION(8,10,8,60,'') ;
             CALL DISPLAY('confirm.program',response) ;
             IF response = 'Y'
             THEN CALL EXECCMDI('DELETE','NOEXEC') ;
          END ;

   ERROROFF _ALL_ ;
   IF WORD(1,'U') =: 'DUP' AND archive ='X'
   THEN DO ;
             ERRORON _ALL_ ;
             _MSG_ ="An ARCHIVED observation
                     cannot be duplicated" ;
          END ;
```

```
   ERROROFF wohb woco woloc woserial ;
   DO i = 1 TO DIM(woparts) ;
        IF FIELD('MODIFIED',woparts(i))
        THEN DO ;
/* the parts of the work order are needed, as well as the whole
number, as well as the first eleven characters */
            wo = LEFT(COMPRESS(wohb||woco||woloc||woserial)) ;
            wo11 = LEFT(COMPRESS(woco||woloc||woserial)) ;

  /* this is one technique to perform a real-time check for
duplicate key field values – another method uses the FETCHOBS
function after the WHERE clause */
            rc = WHERE(dsiself,"wo11=' "||wo11||" ' ") ;
            IF ATTRN(dsiself,'ANY') GT 0
            THEN DO ;
                    ERRORON wohb woco woloc woserial ;
                    CURSOR wohb ;
                    _MSG_ ="This is a duplicate WO number" ;
                END ;
              END ;   * FIELD('MODIFIED',woparts(i)) ;
   END ;   * i = 1 TO DIM(woparts) ;

   DO f = 1 TO 15 ;   /* for each of the ferc variables */
      IF INDEX(woferc(f),'?') > 0 THEN
         DO ;
            dsiferc = OPEN('library.woferc','i') ;
            CALL WREGION(3,15,20,65,") ;

  /* use the data set which is the basis for the format as the
lookup table */
            woferc(f)=
               DATALISTC(dsiferc,'start label','FERC','Y',1) ;
            rc= CLOSE(dsiferc) ;
         END ;
      ELSE IF FIELD('MODIFIED','woferc'||LEFT(PUT(f,2.)))
                AND woferc(f) NE '' THEN
             DO ;
                dsitest= OPEN('library.woferc(WHERE=
                                        (start=" '||woferc(f)||" '))') ;
                IF ATTRN(dsitest,'ANY') GT 0 THEN
                   DO ;
                      woferc(f)= LEFT(COMPRESS(woferc(f),'.')) ;
                      wo19x(f) = TRIM(LEFT(wo11||woferc(f))) ;
                   END ;
                ELSE _MSG_="That FERC is not defined
                                    in the table" ;
/* at this point, the user would enter the word 'FERC' on the
command line in order to add the new ferc value (see code in
next column) */
                rc= CLOSE(dsitest) ;
             END ;

  /* this is delectable --- this is real-time summarization from the
task/invoice data set for this specific ferc number */
     IF cmdword EQ 'REFRESH'  AND woferc(f) NE '' THEN
       DO ;
          dsitask= OPEN('faclitys.taskinvc(WHERE=
                                        (wo19="'||wo19x(f)||'"))');
          IF ATTRN(dsitask,'ANY') GT 0 THEN
             DO ;
                rc= VARSTAT(dsitask,'committd','SUM',commit(f)) ;
                rc= VARSTAT(dsitask,'total','SUM',invctmp1) ;
             END ;
          ELSE invctmp1 = 0 ;
          rc= CLOSE(dsitask) ;
          dsilpocc= OPEN('faclitys.lpocc(WHERE=
                                        (wo19="'||wo19x(f)||'"))');
          IF ATTRN(dsilpocc,'ANY') GT 0
          THEN rc=VARSTAT(dsilpocc,'amount','SUM',invctmp2) ;
          ELSE invctmp2 = 0 ;
          rc= CLOSE(dsilpocc) ;
          invoic(f) = SUM(invctmp1, invctmp2) ;
       END ;
```

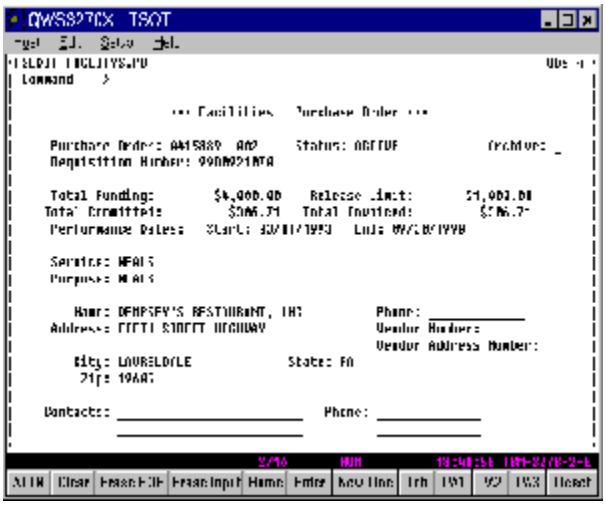
```
 /* this allows the user to jump to the specific task/invoice details -
-- after typing 'TSKINV' on the command line and placing the
cursor on a ferc field and then pressing [ENTER]  */

     IF cmdword EQ 'TSKINV'
        AND CURFLD() EQ 'woferc'||LEFT(PUT(f,2.))
     THEN DO ;
             dsitest =OPEN('faclitys.taskinvc(WHERE=
                                (wo19=" ' ||wo19x(f)|| ' "))');
             IF ATTRN(dsitest,'ANY') GT 0
             THEN CALL FSEDIT(
                     'faclitys.taskinvc(WHERE=
                                        (wo19=" ' ||wo19x(f)|| ' "))',
                     'affspscl.budscrns.taskinvc.screen') ;
             ELSE _MSG_="There are no TASKs/INVOICEs
                                with that WO and FERC" ;
             rc= CLOSE(dsitest) ;
          END ;

   END ;   * FINALLY!, DO f = 1 TO 15 ;

   bdgtttl = SUM(OF budget1-budget15) ;
   cmmtttl = SUM(OF commit1-commit15) ;
   invcttl = SUM(OF invoic1-invoic15) ;

   /* the following snippets, allow the user to jump to the other
data sets.  BE CAREFUL! … each of these opens a distinct
interface, from which the user can jump to another data set ---
yes, even back to here.  It can be very easy to get lost in the
open windows */

    IF WORD(1,'U') EQ 'POX'
    THEN CALL FSEDIT('faclitys.po',
                        'affspscl.budscrns.po.screen') ;

    IF WORD(1,'U') EQ 'TSKINV'
    THEN DO ;
            dsitest =OPEN('faclitys.taskinvc(
                        WHERE=(wo11=" ' ||wo11|| ' "))') ;
            IF ATTRN(dsitest,'ANY') GT 0
            THEN CALL FSEDIT(
                    'faclitys.taskinvc(WHERE=(wo11="'||wo11||'"))',
                    'affspscl.budscrns.taskinvc.screen') ;
            ELSE _MSG_="There are no TASKs/INVOICEs
                                with that WO" ;
            rc= CLOSE(dsitest) ;
         END ;

    IF WORD(1,'U') EQ 'LPOCC'
   THEN DO ;
            dsitest= OPEN('faclitys.lpocc
                        (WHERE=(wo11="'||wo11||'"))') ;
            IF ATTRN(dsitest,'ANY') GT 0
            THEN CALL FSEDIT(
                    'faclitys.lpocc(WHERE=(wo11="'||wo11||'"))',
                    'affspscl.budscrns.lpocc.screen') ;
            ELSE _MSG_="There are no LPOCCs with that WO" ;
            rc= CLOSE(dsitest) ;
         END ;

    IF WORD(1,'U') EQ 'FERC'
    THEN DO ;
            CALL FSVIEW('library.woferc','ADD') ;
         END ;
RETURN ;


TERM:
RETURN ;


FSETERM:
   rc = CLOSE(dsiself) ;
RETURN ;
```

Here is how the lookup window for ferc codes appears:



Purchase Order

A purchase order has no direct tie to a work order; however a task/invoice, which is tied to a work order can only be created from a purchase order screen.



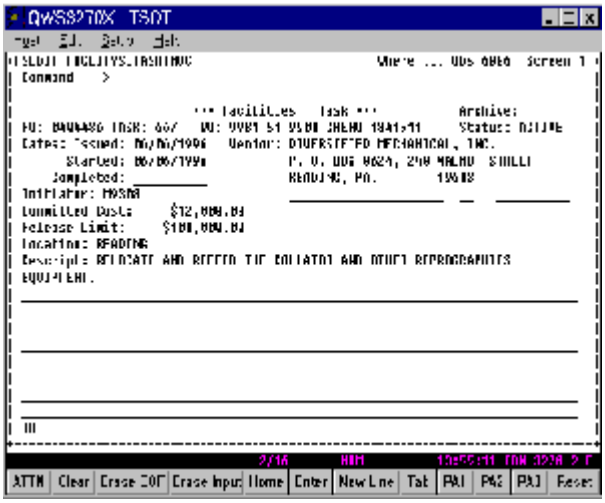
```
/* MSMS.SASPROG.PERM.SAS(SCLPO) */
LENGTH response $ 1 ;   * for the DELETE confirmation window ;

FSEINIT:
  dsiself = OPEN('faclitys.po','i') ;
  dsivdrs = OPEN('faclitys.vendors','i') ;
  CONTROL LABEL ALWAYS ;
RETURN ;

INIT:                    /*  when the user duplicates the observation /*
  ERROROFF po ;     /* to save re-entering most of the data */
  IF OBSINFO('NEW')
  THEN DO ;
            rc = WHERE(dsiself,"po='"||po||"'") ;
            IF ATTRN(dsiself,'ANY') GT 0
            THEN DO ;
                    ERRORON po ;
                   CURSOR po ;
                   _MSG_ ="This is a duplicate PO number" ;
                  END ;
        END ;
RETURN ;

status:
   CALL WREGION(10,35,15,45,'') ;
   IF status IN('?',' ') THEN status = LISTC('status.list','','Y',1) ;
RETURN ;

archive:
   IF archive NE '' THEN archive ='X' ;
RETURN ;


MAIN:
   cmdword = WORD(1,'U') ;   * command line,first word ;
   IF cmdword =: 'DEL'
   THEN DO ;
             response = 'N' ;
            CALL WREGION(8,10,8,60,'') ;
            CALL DISPLAY('confirm.program',response) ;
            IF response = 'Y'
            THEN CALL EXECCMDI('DELETE','NOEXEC') ;
          END ;

   ERROROFF _ALL_ ;
   IF WORD(1,'U') =: 'DUP' AND archive ='X'
   THEN DO ;
            ERRORON _ALL_ ;
            _MSG_ ="An ARCHIVED observation
                    cannot be duplicated" ;
          END ;

   ERROROFF po ;
   IF FIELD('MODIFIED','po')
   THEN DO ;
             rc = WHERE(dsiself,"po='"||po||"'") ;
             IF ATTRN(dsiself,'ANY') GT 0
             THEN DO ;
                     ERRORON po ;
                     CURSOR po ;
                     _MSG_ ="This is a duplicate PO number" ;
                   END ;
          END ;

/* vendor information:  user enters part of a vendor's name and
presses [ENTER]; this code looks in the data set which was
populated using SAS/ACCESS®  */
   _MSG_ = ' ' ;
   rc = FIELD('ERROROFF','vndrname') ;
   IF FIELD('MODIFIED','vndrname') AND vndrname NE ' '
   THEN DO ;
         rc = WHERE(dsivdrs,
                    'vndrname CONTAINS " ' || vndrname || ' "') ;

       /* test for ANY observations */
          IF FETCHOBS(dsivdrs,1) = -1
          THEN DO ;   /* remove the WHERE clause */
                    rc = WHERE(dsivdrs) ;
                    _MSG_ = "NO matches found on the
                                vendor address table!";
                    rc = FIELD('CURSOR','vndrname') ;
                  END ;

       /* test for more than one obs */
          ELSE IF FETCHOBS(dsivdrs,2) = -1
                 THEN DO ;
                        rc = FETCHOBS(dsivdrs,1);
   vndrname = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndrname'));
    vndrnmbr = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndrnmbr')) ;
      vndraddr = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndraddr')) ;
     vndradd1 = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndradd1'));
    vndradd2 = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndradd2')) ;
        vndrcity = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndrcity')) ;
           vndrst = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndrst')) ;
         vndrzip = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndrzip')) ;
                        END ;
```

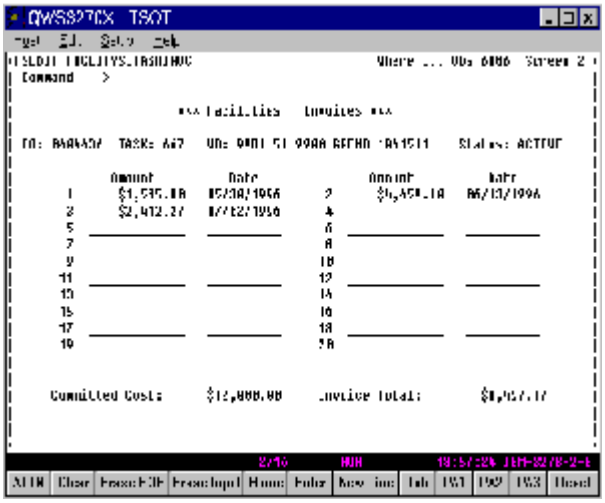
```
            ELSE DO ;   /*get selection from list of matches */
                    CALL WREGION(10,1,24,76);
                     vndrname = DATALISTC(dsivdrs,
                                  'vndrname vndradd1 vndrcity',
                                  'Vendor Names','Y') ;
                     noteid = NOTE(dsivdrs) ;
                     rc = POINT(dsivdrs,noteid) ;
                     rc = FETCH(dsivdrs) ;
  vndrnmbr = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndrnmbr')) ;
   vndraddr = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndraddr')) ;
  vndradd1 = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndradd1')) ;
  vndradd2 = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndradd2')) ;
      vndrcity = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndrcity')) ;
         vndrst = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndrst')) ;
       vndrzip = GETVARC(dsivdrs,VARNUM(dsivdrs,'vndrzip')) ;
          END ;
    END ;

 /* those snippets to jump to the other data sets
maybe these are like the fresh bread or rolls at dinner, they are
always available and if you indulge, you will pay the price  */
   IF WORD(1,'U') EQ 'WO'
   THEN DO ;
             CALL FSEDIT('faclitys.wo',
                         'affspscl.budscrns.wo.screen') ;
         END ;

   IF WORD(1,'U') EQ 'TSKINV'
   THEN DO ;
            dsitest =OPEN('faclitys.taskinvc(
                          WHERE=(po="'||po||'"))') ;
            IF ATTRN(dsitest,'ANY') GT 0
            THEN CALL FSEDIT(
                        'faclitys.taskinvc(WHERE=(po="'||po||'"))',
                        'affspscl.budscrns.taskinvc.screen') ;
            ELSE _MSG_="There are no TASKs/INVOICEs
                          with that PO" ;

    /* I left this line in to show that you could allow the entire data
set to be edited
            ELSE CALL FSEDIT('faclitys.taskinvc',
                         'affspscl.budscrns.taskinvc.screen') ;*/
            rc= CLOSE(dsitest) ;
         END ;

  /* this is how a task/invoice is created --- macro variables are
created to hold values that must be passed to the task/invoice,
then a CALL FSEDIT is performed with the ADD option */
    IF WORD(1,'U') EQ 'XTASK'
   THEN DO ;
             CALL SYMPUT('po',po) ;
             CALL SYMPUT('status',status) ;
             CALL SYMPUTN('release',release) ;
             CALL SYMPUT('vndrname',vndrname) ;
             CALL SYMPUT('vndradd1',vndradd1) ;
             CALL SYMPUT('vndradd2',vndradd2) ;
             CALL SYMPUT('vndrcity',vndrcity) ;
             CALL SYMPUT('vndrst',vndrst) ;
             CALL SYMPUT('vndrzip',vndrzip) ;
             CALL SYMPUTN('vndrphon',vndrphon) ;
             CALL FSEDIT('faclitys.taskinvc',
                         'affspscl.budscrns.taskinvc.screen',
                         'ADD') ;
         END ;

   IF WORD(1,'U') EQ 'LPOCC'
   THEN CALL FSEDIT('faclitys.lpocc',
                    'affspscl.budscrns.lpocc.screen') ;

   IF WORD(1,'U') EQ 'FERC'
   THEN CALL FSVIEW('library.woferc','BROWSE','','BRONLY') ;

RETURN ;
```

```
TERM:
RETURN ;


FSETERM:
   IF dsivdrs > 0 THEN rc = CLOSE(dsivdrs) ;
   rc = CLOSE(dsiself) ;
RETURN ;
```

Task / Invoice

The unique thing to savor with the SCL of task/invoices is how a new one is created; it is required that there be a purchase order first, before any task/invoice can be created. Once this relationship exists, it is possible to duplicate a task/invoice, however the integrity of the system depends upon this initial link.






```
/* MSMS.SASPRORG.PERM.SAS(SCLTSKIN) */

LENGTH response $ 1 ;   * for the DELETE confirmation window ;

FSEINIT:
   dsiself = OPEN('faclitys.taskinvc','i') ;
   CONTROL LABEL ALWAYS ;
/* remind these hard-working people how to get to where they
want to go */
   _MSG_="          * * * INVOICEs are on Screen 2 <F11> * * *" ;

RETURN ;
```

```
INIT:
   LENGTH lasttask 8   cmdword $8 ;
   cmdword = WORD(1,'U') ;   * command line,first word ;

/* Existing tasks/invoices CAN be duplicated in order to save re-
keying data.  A NEW task/invoice must come from a purchase
order therefore . . . */

   IF OBSINFO('NEW') AND cmdword NOT IN('DUP','ADD')
   THEN DO ;
              po = SYMGET('po') ;   /* from PO */

              IF dsiself > 0 THEN rc= CLOSE(dsiself) ;
              dsiself = OPEN('faclitys.taskinvc(
                              WHERE=(po="'||po||'"))','i') ;

  /* calculate the next sequential task number */
              rc = VARSTAT(dsiself,'task','MAX',lasttask) ;
              task = MAX(lasttask + 1,1) ;
              potask = po||PUT(task,Z3.) ;

  /* immediately SAVE the observation so that that NEW task
number is available for another user */
              CALL EXECCMDI('SAVE') ;

              status = SYMGET('status') ;     /* from PO */
              release = SYMGETN('release') ;
              vndrname = SYMGET('vndrname') ;
              vndradd1 = SYMGET('vndradd1') ;
              vndradd2 = SYMGET('vndradd2') ;
              vndrcity = SYMGET('vndrcity') ;
              vndrst = SYMGET('vndrst') ;
              vndrzip = SYMGET('vndrzip') ;
              vndrphon = SYMGETN('vndrphon') ;

  /* fill in these fields automatically */
              taskstrt = DATE() ;
              initator = SYMGET('sysuid') ;

  /* once created the work order number are protected, since this
is a new observation, open the fields for input */
              UNPROTECT wohb woco woloc woserial woferc ;
              CURSOR wohb ;
          END ;

 /*  re-initialize the fields that are unique to each task/invoice */

   ARRAY invdatx(*) invdat1-invdat15 ;
   ARRAY invamtx(*) invamt1-invamt15 ;
   IF OBSINFO('NEW')  AND  WORD(1,'U') =: 'DUP'
   THEN DO ;
              DO i = 1 TO 15 ;
                   IF invdatx(i) NE . THEN invdatx(i) = . ;
                   IF invamtx(i) NE . THEN invamtx(i) = . ;
              END ;
              committd = . ;
              invoiced = . ;
              UNPROTECT woferc ;
              CURSOR woferc ;
          END ;
RETURN ;
status:
   CALL WREGION(10,35,15,45,'') ;
   IF status IN('?',' ')
   THEN status = LISTC('status.list','','Y',1) ;
RETURN ;

woco:
   CALL WREGION(10,35,15,45,'') ;
   IF woco IN('?',' ')
   THEN woco = LISTC('company.list','','Y',1) ;
RETURN ;

woferc:
   IF INDEX(woferc,'?') > 0
   THEN DO ;
              dsiferc = OPEN('library.woferc','i') ;
              CALL WREGION(3,15,20,65,'') ;
              woferc=DATALISTC(dsiferc,'start label','FERC','Y',1);
               rc= CLOSE(dsiferc) ;
         END ;
   ELSE DO ;
              dsitest =OPEN('library.woferc(
                              WHERE=(start="'||woferc||'"))') ;
              IF ATTRN(dsitest,'ANY') GT 0
              THEN ;  /* A FERC is defined */
              ELSE _MSG_="That FERC is not defined
                              in the table" ;
              rc= CLOSE(dsitest) ;
        END ;

              /* remove any spaces, commas, and underlines */
    woferc = LEFT(COMPRESS(woferc,'.,_')) ;
RETURN ;


MAIN:
   IF cmdword =: 'DEL'
   THEN DO ;
              response = 'N' ;
              CALL WREGION(8,10,8,60,'') ;
              CALL DISPLAY('confirm.program',response) ;
              IF response = 'Y'
              THEN CALL EXECCMDI('DELETE','NOEXEC') ;
         END ;

/* the MODIFIED function only allows three field names, so. . . */
   IF FIELD('MODIFIED','wohb')
      OR  FIELD('MODIFIED','woco')
      OR  FIELD('MODIFIED','woloc')
      OR  FIELD('MODIFIED','woserial')
      OR  FIELD('MODIFIED','woferc')
   THEN DO ;  /* create the non-displayed variable that link this
                 task/invoice to a work order */
              wo=LEFT(COMPRESS(
                        wohb||woco||woloc||woserial||woferc,'.,_'));
              wo19=LEFT(COMPRESS(
                          woco||woloc||woserial||woferc,'.,_'));
             wo11=LEFT(COMPRESS(woco||woloc||woserial,'.,_'));

               dsitest =OPEN('faclitys.wo(
                              WHERE=(wo11="'||wo11||'"))') ;
              IF ATTRN(dsitest,'ANY') GT 0
              THEN ;   /*A work order exists for this task/invoice */
              ELSE _MSG_="There are no WOs with that
                              WO(CO/LOC/SERIAL)" ;
               rc= CLOSE(dsitest) ;
         END ;

   invoiced = SUM(OF invamt1-invamt20) ;

   ERROROFF _ALL_ ;
   IF WORD(1,'U') =: 'ADD'
   THEN DO ;   /* if the user attempts to ADD a new obs */
              PROTECT _ALL_ ;
              ERROROn _ALL_ ;
              HOME ;
              _MSG_ ="To ADD a TASK, start at PO screen" ||
                        " and type XTASK - " <F2> to Cancel" ;
         END ;
   ERROROFF _ALL_ ;
   IF WORD(1,'U') =: 'DUP' AND archive ='X'
   THEN DO ;     ERROROn _ALL_ ;     CURSOR po ;
              _MSG_ ="An ARCHIVED observation cannot
                        be duplicated" ;
         END ;
```

```
  /*last chance for those snippets to jump to the other data sets */
IF WORD(1,'U') EQ 'WO'
   THEN DO ;
             dsitest =OPEN('faclitys.wo(
                                WHERE=(wo11="'||wo11||'"))') ;
             IF ATTRN(dsitest,'ANY') GT 0
             THEN CALL FSEDIT(
                        'faclitys.wo(WHERE=(wo11="'||wo11||'"))',
                        'affspscl.budscrns.wo.screen') ;
             ELSE _MSG_="There are no WOs with that WO" ;
          /* ELSE CALL FSEDIT('faclitys.wo',
                                'affspscl.budscrns.wo.screen');*/
             rc = CLOSE(dsitest) ;
          END ;

   IF WORD(1,'U') EQ 'POX'
   THEN DO ;
             dsitest =OPEN('faclitys.po(WHERE=(po="'||po||'"))') ;
             IF ATTRN(dsitest,'ANY') GT 0
             THEN CALL FSEDIT(
                        'faclitys.po(WHERE=(po="'||po||'"))',
                        'affspscl.budscrns.po.screen') ;
             ELSE _MSG_="There are no POs with that PO" ;
           /* ELSE CALL FSEDIT('faclitys.po',
                                'affspscl.budscrns.po.screen');*/
             rc = CLOSE(dsitest) ;
          END ;

   IF WORD(1,'U') EQ 'LPOCC'
   THEN CALL FSEDIT('faclitys.lpocc',
                        'affspscl.budscrns.lpocc.screen') ;

   IF WORD(1,'U') EQ 'FERC'
   THEN CALL FSVIEW('library.woferc','BROWSE','','BRONLY') ;

RETURN ;


TERM:
RETURN ;


FSETERM:
  IF dsiself > 0 THEN rc= CLOSE(dsiself) ;
RETURN ;
```

And finally . . .

The following program is the one executed nightly to summarize amounts for each work order and synchronize the status of tasks, purchase orders, and work orders. This one was tricky. Take a deep breath and let out an 'aaawww'; I am almost done.

```
/*   MSMS.SASPORG.PERM.SAS(WOCMTACT)  */

PROC MEANS DATA= faclitys.taskinvc   SUM NOPRINT ;
   VAR committd invoiced ;
   CLASS wo19 ;
   OUTPUT OUT= tskinv   SUM= ;
RUN ;

PROC MEANS DATA= faclitys.lpocc   SUM  NOPRINT ;
   VAR amount ;
   CLASS wo19 ;
   OUTPUT OUT= lpocc   SUM= ;
RUN ;



          /* the DROPed fields are being recalculated */
DATA wo (DROP= wo19_1-wo19_15  woferc1-woferc15
              budget1-budget15 commit1-commit15 invoic1-invoic15
                    bdgtttl cmmtttl invcttl budget  wo19)
       wofrcbud (KEEP= wo11 wo19 budget)
       ;
   SET faclitys.wo ;
   OUTPUT wo ;

   ARRAY wo19x(15) $ wo19_1-wo19_15 ;
   ARRAY wofercx(15) $ woferc1-woferc15 ;
   ARRAY budgetx(15) $ budget1-budget15 ;
   DO i = 1 TO 15 ;     /* keep only those fields that have values */
      IF wofercx(i) ne ' '
      THEN DO ;
                wo19 = wo19x(i) ;
                budget = budgetx(i) ;
                OUTPUT wofrcbud ;
           END ;
   END ;
   DROP i ;     /* Do not forget to do this !!! */
RUN ;

PROC SQL ;
   CREATE TABLE tskinlpo AS
   SELECT t.wo19, t.committd,
             SUM(t.invoiced, l.amount) AS invoicex
   FROM tskinv t  LEFT JOIN  lpocc l
          ON t.wo19 = l.wo19
          AND t._TYPE_ GT 0  AND l._TYPE_ GT 0
   ;
   CREATE TABLE wotskinv AS
   SELECT w.wo19, w.wo11, w.budget, ti.committd,
             ti.invoicex AS invoiced
   FROM wofrcbud w  LEFT JOIN  tskinlpo ti
          ON w.wo19 = ti.wo19
   UNION
   SELECT ti.wo19, wo11, w.budget, ti.committd,
             ti.invoicex AS invoiced
   FROM wofrcbud w  RIGHT JOIN  tskinlpo ti
          ON w.wo19 = ti.wo19
   ORDER BY wo11, wo19
   ;
QUIT ;
RUN ;

        /* Bring the pieces back together again … */
DATA qwerty (KEEP= wo11 wo19_1-wo19_15  woferc1-woferc15
                   budget1-budget15 commit1-commit15
                   invoic1-invoic15)  ;
   ARRAY wo19x(15)  $19.  wo19_1-wo19_15 ;
   ARRAY wofercx(15)  $8.  woferc1-woferc15 ;
   ARRAY budgetx(15)  8.  budget1-budget15 ;
   ARRAY commitx(15)  8.  commit1-commit15 ;
   ARRAY invoicx(15)  8.  invoic1-invoic15 ;

   DO i = 1 TO 15 UNTIL(LAST.wo11) ;
      SET wotskinv (WHERE=
                   (SUM(budget,committd,invoiced) NE .)) ;
      BY wo11 ;
      IF wo11 NE ''
      THEN DO ;
                wo19x(i) = wo19 ;
                wofercx(i) = SUBSTR(wo19,12,8) ;
                budgetx(i) = budget ;
                commitx(i) = committd ;
                invoicx(i) = invoiced ;
                IF LAST.wo11  AND  wo11 NE ''
               THEN OUTPUT ;
           END ;
   END ;
   DROP i wo19 budget committd invoiced ;
RUN ;
```

```
PROC SORT DATA= wo ;
    BY wo11 ;
RUN ;

DATA woqwerty ;
   MERGE wo (IN= w)  qwerty (IN= q) ;
   BY wo11 ;
   IF w ;

   bdgtttl =SUM(OF budget1-budget15) ;
   cmmtttl =SUM(OF commit1-commit15) ;
   invcttl =SUM(OF invoic1-invoic15) ;

   FORMAT budget1-budget15 commit1-commit15
          invoic1-invoic15 bdgtttl cmmtttl invcttl
          DOLLAR15.2;
RUN ;

PROC SORT DATA= woqwerty
            OUT= faclitys.wo (INDEX=(woserial wo/UNIQUE
                                     wo11/UNIQUE)) ;
   BY archive wo11 ;
RUN ;
```

CONCLUSION

Whew! Are you full yet? This paper has only been a taste-testing (okay, it was a feast) of the cornucopia of tools the SAS system provides to create truly exquisite data entry applications. I trust I have whetted your appetite with these morsels and that you will incorporate at least some of these recipes the next time your client asks you to provide the catering. For those of you who have not ventured into using SCL(whether you call it Screen Control Language or SAS® Component Language) with your FSEDIT applications, let me just encourage you to jump into the kettle and try it; you will like it as much as this Michael did, and your clients will give you a standing ovation and maybe even some dessert.
Bon appetite !

REFERENCES

SAS Institute Inc. (1990), *SAS Guide to Macro Processing, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1989), *SAS/ACCESS® INTERFACE to DB2™: Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1989), *SAS/AF ®Software: Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1989), *SAS/FSP ®Software: Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1990), *SAS ®Guide to Macro Processing, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1990), *SAS ®Procedures Guide, Version 6, Third Edition*, Cary, NC: SAS Institute Inc.
SAS Institute Inc. (1994), *SAS ® Screen Control Language: Reference, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1991), *SAS/SHARE ®Software: Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1991), *SAS ®Technical Report P-222, Changes and Enhancements to Base SAS ® Software, Release 6.07*, Cary, NC: SAS Institute Inc.

Michael A. Mace
1152 E 344 ST
Eastlake, OH 44095-2942
440-953-8924 (H)
330-796-3981 (W)
e-mail: **michael.a.mace@worldnet.att.net**