Paper 223

Using SAS/FSP® Software and a Two-Dimensional Array to Manage Distribution of Monthly Production Reports

Helen-Jean Talbott, Commercial Credit Corporation, Baltimore, MD

ABSTRACT

Within the first three days of each month, the Credit Policy MIS group in Commercial Credit produces over 200 reports which must be distributed quickly to more than 70 recipients. Until recently, distribution information was kept in a simple Microsoft Excel® spreadsheet showing report numbers, program names, report titles, recipient names, and number of copies for each recipient. Although the spreadsheet approach worked well when there were 20 recipients, it became impractical with more than 40 recipients due to space limitation. The goal was to use SAS/FSP® to manage the distribution data and to use this data to produce check-off lists. Structure of the data set was key to success. To make the SAS/FSP application run smoothly, we wanted one observation per program containing information on all the reports and recipients for that program. A two-dimensional array accommodating up to 32 reports and 20 recipients per program worked very well. The tricky part was creating a SAS program to populate the two-dimensional array based on the data derived from the Excel spreadsheet. The purpose of this paper is to show: (1) creation of the SAS two-dimensional array, (2) development of the SAS/FSP application, and (3) program used to produce check-off lists.

INTRODUCTION

The computer print room for Commercial Credit is an extremely busy place the first five calendar days of each month. The largest portion of production reports for most departments is produced monthly at the beginning of the month. Printing these reports requires a tremendous amount of paper. One of these departments, Credit Policy, produces more than 200 production reports in the first three days and these must be distributed quickly to more than 70 recipients. This individual distribution changes frequently. The number of recipients depends on the specific report number.

Until recently, distribution information was kept on a personal computer in a simple Microsoft Excel® spreadsheet showing report numbers, program names, report titles, recipient names, and number of copies for each recipient. There was one row per report number. Columns included the program name and report title and a separate column for each of the common recipients. Values within the spreadsheet were either a number (for the number of copies) or an “X”. One difficult feature of this spreadsheet was that special recipients (not included among the common recipients) were coded by marking the “MISC” column with the number of copies needed, and then listing the names of special recipients in the report title column, after the title text. Check-off lists printed from this data were in order of the report number. Unfortunately, report numbers within a program frequently skip around, making it hard to locate all the reports for one program on the list. Although the spreadsheet approach worked well when there were fewer report numbers and only 18 recipients, it became impractical with more than 40 recipients.

The goal was to build a new system using SAS/FSP to manage the distribution data (program names, report numbers, titles, recipients for each report) and to use this data to produce check-off lists. Initial work on this problem showed that it would be a challenge to create the initial data set. The starting point was the Excel data with one observation per report number. What we needed was one observation per production program containing information on all the reports and recipients for that program so that the Information is “packaged” the same way as the stacks of printed paper.

The solution to the programming problem was a two-dimensional array accommodating up to 32 reports and up to 20 recipients per program. The tricky part was creating a SAS program to populate the two-dimensional array based on the information in the spreadsheet. These programs were developed using SAS version 6.12 on a personal computer running under Windows NT 4.0.

The purpose of this paper is to show: (1) creation of the SAS two-dimensional array, (2) development of the SAS/FSP application, and (3) programs used to produce check-off lists.

BUILDING SAS DATA SET FROM EXCEL SPREADSHEET

The first step in this process is to import the data from Excel. Use the SAS Import Wizard in SAS version 6.12, which automatically codes the necessary SAS/ACCESS® statements to create a SAS data set (called RPTLOG1) from the EXCEL file. Each of the columns of the Excel file becomes a variable in the SAS data set. There is one observation per report number and one production program can produce several report numbers. There are also many blank lines in the Excel file.

For this application, the SAS data set RPTLOG1 contains the following variables. To save space, we will show only 5 recipient names.

```
#   Variable   Type   Len   Pos   Format

20  ABERNATH   Num      8   351   3.
10  BBENNER    Num      8   271   8.
11  BRIANM     Num      8   279   8.
21  BROOKE     Num      8   359   3.
    (**etc.**)
 5  CREDITBK   Num      8   231   2.
 2  PROGRAM    Char    16    11   $16.
 1  REPORTNO   Char    11     0   $11.
 3  TITLE      Char   200    27   $200.
25  VAR24      Char     1   391   $1.
26  VAR25      Char     8   392   $8.
27  VAR26      Char     8   400   $8.
    (**etc.**)
144 VAR143     Char     8  1336   $8.
```

The next step is to remove unwanted variables VAR24 - VAR143 that were empty cells in the original Excel spreadsheet. Use the following simple SAS program:

```
LIBNAME IN1 'C:\MISRPT';
DATA IN1.RPTLOG2;
   SET RPTLOG1(DROP=VAR24-VAR143);
RUN;
```

Now the next program can be run which creates the initial data set for the SAS/FSP application. There are 18 common

recipients in the original Excel file (that will become temporary data sets a-r). Again, code for the first 5 recipients is shown. Some of this repetitive code could have been placed in macros, but this not a production program and the amount of coding is fairly small. There are many comments within the program that describe how it operates. Remember that the overall purpose of this program is to produce a SAS data set with one observation per production program and with the variables in the proper sequence for the SAS/FSP screen.

This program uses several arrays. The array "person" has 20 elements (one for each of 20 possible recipients), and holds the names of the people that receive any report numbers produced by that particular production program. The array "rpt" has 32 elements, and holds up to 32 report numbers that are produced by the production program. Arrays "rtitle" and "copies" hold the report titles and total number of copies for each of the report numbers within the production program.

The main feature of this program is the array "pr" which is a two-dimensional array that holds the number of copies needed for each person-report combination within the program. As described above, there is provision for 20 people and 32 reports within a program. Thus, array element "p1r1" holds the number of copies needed for person number 1, report number 1, and element "p1r32" holds the number of copies needed for person number 1, report number 32. Later, when this information is displayed in the SAS/FSP screens, the people (array "person") become the row labels of a table, the report numbers (array "rpt") become the column labels of the table, and the number of copies (array "pr") become the values in the interior of the table.

Processing the data for the first report number in a production program is easy. Let's say that there are 3 people (Brooke, Scott, and Mary) that receive the first report number (H237) in production program BCOP3DAY. Mary should get 2 copies and Brooke and Scott each get 1 copy. The variables, or elements of the arrays, would have the following values:

Variable	value
program	BCOP3DAY
rpt1	H237
person1	BROOKE
person2	SCOTT
person3	MARY
p1r1	1
p2r1	1
p3r1	2

Subsequent report numbers are more difficult to process because you have to determine whether the person is already receiving another report number in the same program, and then determine which elements of the array "pr" should be populated. Let's say that the next report number in program BCOP3DAY is H238 and that Mary gets 2 copies and Winston gets 1 copy. Our program needs to determine that for this production program (BCOP3DAY) Mary is person number 3 and that element p3r2 should get a value of 2. Winston is not in the list of people that get the first report (h237 above), so person4 gets the value "WINSTON" and element p4r2 is set to 1.

This process continues until all of the report numbers within the production program have been processed. Then the observation is output.

```
*******************************************;
* c:\misrpt\makedsn3                      *;
*******************************************;
*                     Helen-Jean Talbott  *;
*                     10/16/98            *;
*                                         *;
```

```
* create initial data set for MIS report     *;
* tracking based on data imported from EXCEL *;
*********************************************;
options ls=177 ps=54;
libname in1 'c:\misrpt';

data a(keep=program reportno people amt)
     b(keep=program reportno people amt)
     c(keep=program reportno people amt)
     d(keep=program reportno people amt)
     e(keep=program reportno people amt)
         (**etc.**)
     s(keep=program reportno title);

   length people $ 16
          reportno $ 9
          amt 8;

   set in1.rptlog2;

   if reportno=' ' then delete;
   if people=' ' and title=' ' then delete;

   if abernath >0 then do;
      people='D.ABERNETHY';
      amt=abernath;
      output a;
      end;
   if bbenner >0 then do;
      people='W.BENNER';
      amt=bbenner;
      output b;
      end;
   if brianm >0 then do;
      people='B.MAXWELL';
      amt=brianm;
      output c;
      end;
   if brooke >0 then do;
      people='B.DUVALL';
      amt=brooke;
      output d;
      end;
   if creditbk >0 then do;
      people='CREDITBK';
      amt=creditbk;
      output e;
      end;

   (**etc.**)

   *** report title only ***;
   output s;
run;

proc sort data=a; by program reportno people;
proc sort data=b; by program reportno people;
proc sort data=c; by program reportno people;
proc sort data=d; by program reportno people;
proc sort data=e; by program reportno people;
   (**etc.***)
proc sort data=s; by program reportno;
run;

data combine(drop=people amt i j maxper rcount
                  reportno title);
   set s a b c d e f g h i j k l m n o p q r;
   by program reportno;

   array pr{20,32}
      p1r1  - p1r32  p2r1  - p2r32
      p3r1  - p3r32  p4r1  - p4r32
      p5r1  - p5r32  p6r1  - p6r32
      p7r1  - p7r32  p8r1  - p8r32
      p9r1  - p9r32  p10r1 - p10r32
      p11r1 - p11r32  p12r1 - p12r32
      p13r1 - p13r32  p14r1 - p14r32
      p15r1 - p15r32  p16r1 - p16r32
      p17r1 - p17r32  p18r1 - p18r32
```

```
      p19r1 - p19r32  p20r1 - p20r32;
   array copies{32};
   array person{20} $ 20;
   array rpt{32} $ 9;
   array rtitle{32} $ 200;

   retain i j person maxper rcount rpt rtitle
          copies pr;

   format
   p1r1 - p1r32   p2r1 - p2r32    p3r1 - p3r32
   p4r1 - p4r32   p5r1 - p5r32    p6r1 - p6r32
   p7r1 - p7r32   p8r1 - p8r32    p9r1 - p9r32
  p10r1 - p10r32 p11r1 - p11r32  p12r1 - p12r32
  p13r1 - p13r32 p14r1 - p14r32  p15r1 - p15r32
  p16r1 - p16r32 p17r1 - p17r32  p18r1 - p18r32
  p19r1 - p19r32 p20r1 - p20r32
   copies1-copies32 3.;

   if first.program then do;
      *--------------------*;
      * reset the counters *;
      *--------------------*;
      do j=1 to 32;
         rpt{j}=' ';
         rtitle{j}=' ';
         copies{j}=' ';
         do i=1 to 20;
            pr{i,j}=.;
            person{i}=' ';
            end;
         end;
      i=1;
      j=1;
      maxper=0;
      end;

   if first.reportno then do;
   ***(this record has title only )***;
      *** capture report number ***;
      rpt{j}=reportno;
      *** capture report title ***;
      rtitle{j}=title;
      return;
      end;

   *---------------------------------*;
   * first report within the program *;
   *---------------------------------*;
   if j=1 then do;
      *** initialize # of copies to zero ***;
      if i=1 then rcount=0;
      *** identify the people and # copies ***;
      person{i}=people;
      pr{i,1}=amt;
      *** how many people so far? ***;
      maxper+1;
      *** update total number of copies ***;
      rcount+amt;
      *** last record for this report? ***;
      if last.reportno then go to last;
      *** fetch next record for report ***;
      i+1;
      return;
      end;

   *-----------------------------------------*;
   * additional reports within same program  *;
   *-----------------------------------------*;
   else do;
      *** repeat until find person match  ***;
      *** or add to person list after all ***;
      *** persons have been tried         ***;
      *** if "PEOPLE" is in person list   ***;
      *** then update corresponding cell  ***;

oldper:
      if people=person{i} then do;
         pr{i,j}=amt;
         rcount+amt;
         *** last record for this report? ***;
         if last.reportno then go to last;
         *** fetch the next record ***;
         i=1;
         return;
         end;

      else do;
         if i > maxper then go to newper;
         else do;
            i+1;
            go to oldper;
            end;
         end;


      ***( new person -- update person list and
         update corresponding cell )***;
newper:
      person{i}=people;
      pr{i,j}=amt;
      rcount+amt;
      maxper+1;
      *** last record for this report? ***;
      if last.reportno then go to last;
      *** fetch the next record ***;
      i=1;
      return;
      end;

last:
      *** number of copies for this report ***;
      copies{j}=rcount;
      rcount=0;
      if last.program then do;
         output;
         return;
         end;
      *** increment j counter for next report;
      j+1;
      i=1;
      return;
run;

data in1.rptlog;
   *** this looks redundant, but the order ***;
   *** in the length statement controls    ***;
   *** order of variables on SAS/FSP screen***;
   *** and will save a lot of time later   ***;

   length
      program           $ 16
      rpt1 - rpt8       $ 9
      rtslp1 - rtslp8   $ 1
      person1           $ 20
      p1r1 - p1r8         3
      person2           $ 20
      p2r1 - p2r8         3
      person3           $ 20
      p3r1 - p3r8         3
      person4           $ 20
      p4r1 - p4r8         3
      person5           $ 20
      p5r1 - p5r8         3
      person6           $ 20
      p6r1 - p6r8         3
      person7           $ 20
      p7r1 - p7r8         3
      person8           $ 20
      p8r1 - p8r8         3
      person9           $ 20
      p9r1 - p9r8         3
      person10          $ 20
      p10r1 - p10r8       3
      person11          $ 20
      p11r1 - p11r8       3
      person12          $ 20
      p12r1 - p12r8       3
```

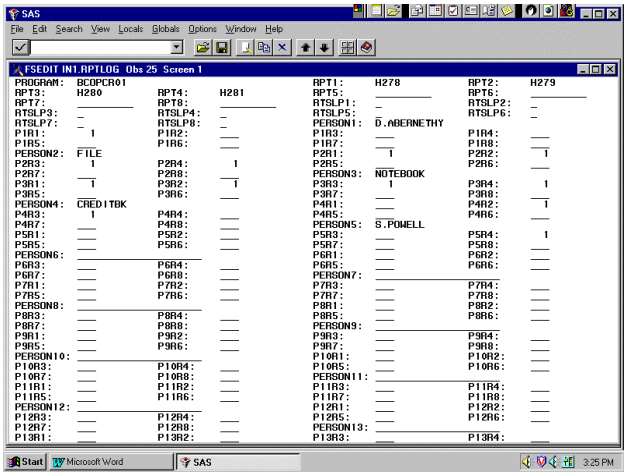
```
      person13          $ 20
      p13r1 - p13r8       3
      person14          $ 20
      p14r1 - p14r8       3
      person15          $ 20
      p15r1 - p15r8       3
      person16          $ 20
      p16r1 - p16r8       3
      person17          $ 20
      p17r1 - p17r8       3
      person18          $ 20
      p18r1 - p18r8       3
      person19          $ 20
      p19r1 - p19r8       3
      person20          $ 20
      p20r1 - p20r8       3
      copies1 - copies8   3
      rtitle1 - rtitle8 $ 200
      rpt9 - rpt16      $ 9
      rtslp9 - rtslp16  $ 1
      p1r9 - p1r16        3
      p2r9 - p2r16        3
      p3r9 - p3r16        3
      p4r9 - p4r16        3
      p5r9 - p5r16        3
      p6r9 - p6r16        3
      p7r9 - p7r16        3
      p8r9 - p8r16        3
      p9r9 - p9r16        3
      p10r9 - p10r16      3
      p11r9 - p11r16      3
      p12r9 - p12r16      3
      p13r9 - p13r16      3
      p14r9 - p14r16      3
      p15r9 - p15r16      3
      p16r9 - p16r16      3
      p17r9 - p17r16      3
      p18r9 - p18r16      3
      p19r9 - p19r16      3
      p20r9 - p20r16      3
      copies9 - copies16  3
      rtitle9 - rtitle16 $ 200
      rpt17 - rpt32     $ 9
      rtslp17 - rtslp32 $ 1
      p1r17 - p1r32       3
      p2r17 - p2r32       3
      p3r17 - p3r32       3
      p4r17 - p4r32       3
      p5r17 - p5r32       3
      p6r17 - p6r32       3
      p7r17 - p7r32       3
      p8r17 - p8r32       3
      p9r17 - p9r32       3
      p10r17 - p10r32     3
      p11r17 - p11r32     3
      p12r17 - p12r32     3
      p13r17 - p13r32     3
      p14r17 - p14r32     3
      p15r17 - p15r32     3
      p16r17 - p16r32     3
      p17r17 - p17r32     3
      p18r17 - p18r32     3
      p19r17 - p19r32     3
      p20r17 - p20r32     3
      copies17 - copies32 3
      rtitle17 - rtitle32 $ 200
      ;
   set combine;

   retain rtslp1 - rtslp32 ' ';
run;
```

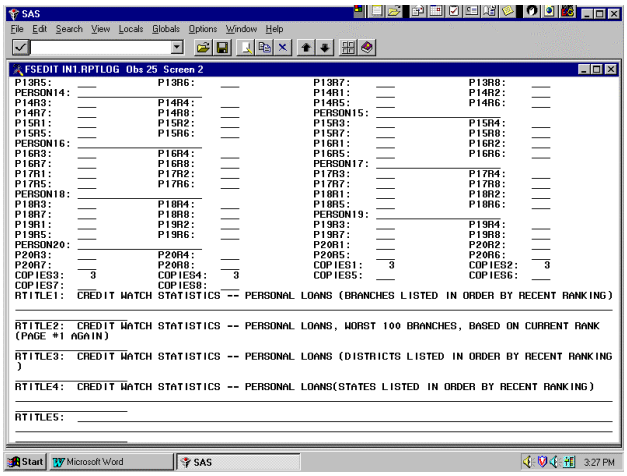
SAS/FSP SCREENS AND SCL CODE

The SAS/FSP session for this application may be started using the following code. The data set (RPTLOG.sd2) and the custom SAS/FSP screen (RPTLOG.sc2) both are located in directory C:\misrpt.

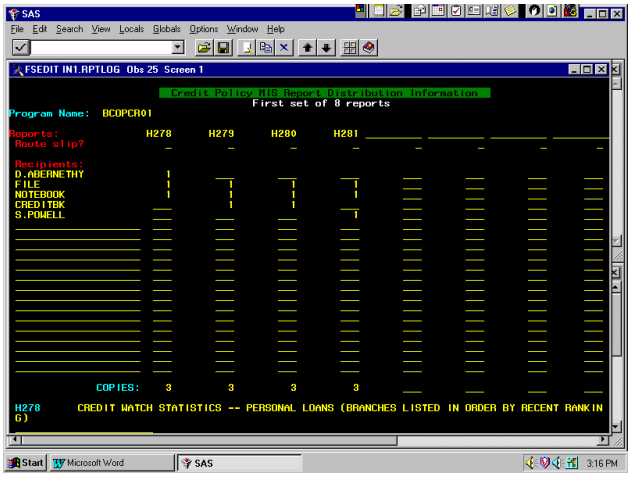
```
libname in1 'c:\misrpt';
proc fsedit data=in1.rptlog screen=in1.rptlog;
run;
```

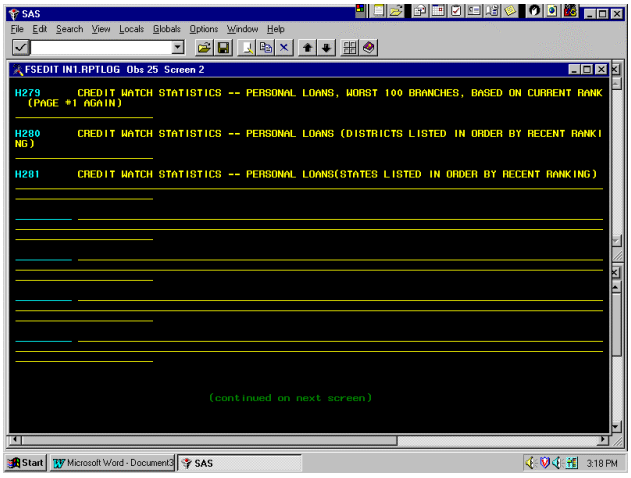
Initially the screens will look similar to the screens shown below. The first screen shows the all the people and copies for the first 8 report numbers. The second screen continues with the titles for these report numbers. There are four sets of 8 report numbers for each program, and two SAS/FSP screens will be used to display information for each set of 8 report numbers. The variables are in the desired order, but their placement on the different lines needs to changed somewhat. Most of the names of the variables will be removed, leaving places for the values only. Change the screen using the FSEDIT Modify and Identify windows in SAS/FSP. You can also change many features of the screen, such as the color of the background, color of the text, and settings for initial values of new observations with the FSEDIT Attribute window.





The final screens for the first set of 8 report numbers are shown below with actual data from the earlier screen. Note that for this program (BCOPCR01) there are only a four report numbers and five recipients.





The following is the SCL code that is part of this custom SAS/FSP screen. Some of the functions of this code are: (1) to recalculate the number of copies required for each report number in the program, (2) to reset the number of copies for all reports if a person is removed from the distribution, and (3) to reset the number of copies and titles if a report number is removed from the program. Notice that the same arrays are used here as for the earlier program.

```
array person{20} $20;
array pr{20,32}
  p1r1 - p1r32    p2r1 - p2r32
  p3r1 - p3r32    p4r1 - p4r32
  p5r1 - p5r32    p6r1 - p6r32
  p7r1 - p7r32    p8r1 - p8r32
  p9r1 - p9r32    p10r1 - p10r32
  p11r1 - p11r32  p12r1 - p12r32
  p13r1 - p13r32  p14r1 - p14r32
  p15r1 - p15r32  p16r1 - p16r32
  p17r1 - p17r32  p18r1 - p18r32
  p19r1 - p19r32  p20r1 - p20r32;
array rpt{32} $9;
array rtitle{32} $ 200;

FSEINIT:
return;

INIT:
return;

MAIN:
*** reset counts if person is deleted ***;
   do i=1 to 20;
      if person{i}=' ' then do;
         do j=1 to 32;
            pr{i,j}=.;
            end;
         end;
      end;
*** reset counts + title if report is deleted;
   do j=1 to 32;
      if rpt{j}=' ' then do;
         rtitle{j}=' ';
         do i=1 to 20;
            pr{i,j}=.;
            end;
         end;
      end;

copies1
= sum( p1r1,  p2r1,  p3r1,  p4r1,  p5r1,
       p6r1,  p7r1,  p8r1,  p9r1, p10r1,
       p11r1, p12r1, p13r1, p14r1, p15r1,
       p16r1, p17r1, p18r1, p19r1, p20r1);

copies2
= sum( p1r2,  p2r2,  p3r2,  p4r2,  p5r2,
       p6r2,  p7r2,  p8r2,  p9r2, p10r2,
       p11r2, p12r2, p13r2, p14r2, p15r2,
       p16r2, p17r2, p18r2, p19r2, p20r2);

copies3
= sum( p1r3,  p2r3,  p3r3,  p4r3,  p5r3,
       p6r3,  p7r3,  p8r3,  p9r3, p10r3,
       p11r3, p12r3, p13r3, p14r3, p15r3,
       p16r3, p17r3, p18r3, p19r3, p20r3);

(**etc.**)

return;

TERM:
return;

FSETERM:
return;
```

After the SAS/FSP screen is complete, it is necessary to edit the data. Under the Excel method, names of special recipients were placed in the same field as the report titles. These special recipients need to be added to the distribution in their proper place and the titles need to be corrected.

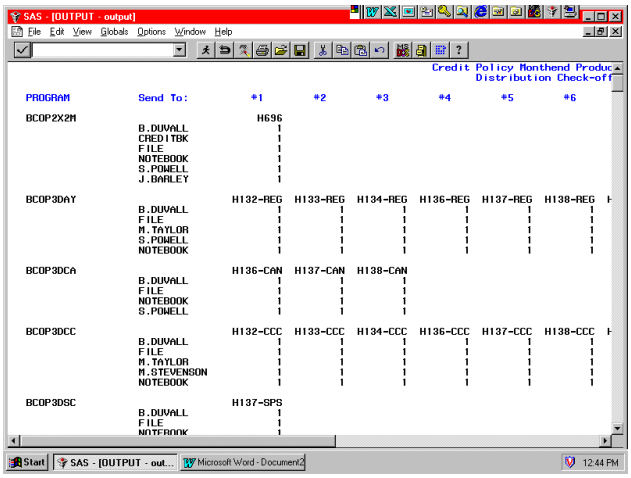
CHECK-OFF LIST REPORT PROGRAM

The distribution check-off list is used to make sure that all reports have been produced every month and that they are delivered to the correct people. Any changes in the distribution are noted and corrected for next month.

The following program produces the check-off list. There are some simplifying assumptions. Few programs produce more that 14 report numbers. Those programs with more than 14 report numbers have a very predictable distribution pattern in that all persons receiving reports from that program receive all the reports for that program. Report titles are not needed for the check-off list, but they are used in other programs related to production reports.

```
*********************************;
* c:\misrpt\checklst.pgm          *;
*********************************;
*               Helen-Jean Talbott*;
```

```
*                10/27/98          *;
*                                  *;
* program to print a check-off list*;
* for monthend production reports  *;
************************************;
options ls=177 ps=54;
libname in1 'f:\taylorm\misrpt';

proc sort data=in1.rptlog
          out=rawdata;
   by program;

data rawdata2(keep=program people
                   report1- report14);
   set rawdata;
   by program;

   length people $20;

   array person{20} $20;
   array rpt{32} $9;
   array report{32} $9;
   array pr{20,32}
     p1r1 - p1r32   p2r1 - p2r32
     p3r1 - p3r32   p4r1 - p4r32
     p5r1 - p5r32   p6r1 - p6r32
     p7r1 - p7r32   p8r1 - p8r32
     p9r1 - p9r32   p10r1 - p10r32
     p11r1 - p11r32 p12r1 - p12r32
     p13r1 - p13r32 p14r1 - p14r32
     p15r1 - p15r32 p16r1 - p16r32
     p17r1 - p17r32 p18r1 - p18r32
     p19r1 - p19r32 p20r1 - p20r32;


   if first.program then do;
      people=' ';
      *** capture 1st 14 reports ***;
      do j=1 to 14;
         if rpt{j} ne ' ' then
            report{j}=right(rpt{j});
         end;
      output;
      i=1;
      end;

   *** info for each recipient ***;
getit:   if person{i} ne ' ' then do;
            people=person{i};
            *** get 1st 14 rpts ***;
            do j=1 to 14;
               if rpt{j} ne ' ' then
               report{j}=pr{i,j};
               end;
            output;
            end;
         i+1;
         if person{i} ne ' ' then
            go to getit;

proc print data=rawdata2 noobs uniform
           split='*';
  by program;
  id program;
  var people
     report1 report2 report3  report4
     report5 report6 report7 report8
     report9 report10 report11
     report12 report13 report14;
  label people='Send To:'
      report1='#1'   report2='#2'
      report3='#3'   report4='#4'
      report5='#5'   report6='#6'
      report7='#7'   report8='#8'
      report9='#9'   report10='#10'
      report11='#11' report12='#12'
      report13='#13' report14='#14';
title1 'Credit Policy Monthend Production';
title2 'Distribution Check-off List';
run;
```



CONCLUSIONS

This paper demonstrates how a two-dimensional array and SAS/FSP can be used to track the monthly distribution of production reports to multiple recipients. The main challenge for developing this system was to create a SAS program that would populate the two-dimensional array based on data in an existing Excel application that had one record per report number and multiple report numbers per production program. Careful design of the SAS data set allowed for easy development of the SAS/FSP screen later in the process. The SAS/FSP screen and SCL code are shown.

REFERENCES

SAS Institute Inc. (1990), SAS Language, Cary, NC: SAS Institute Inc., 1042 pp.

ACKNOWLEDGMENTS

CONTACT INFORMATION

Helen-Jean Talbott
Commercial Credit Corporation
300 St. Paul Place, BSP10A
Baltimore, MD 21202