BATCH PROCESSING AND ADVANCED TOPICS USING PROC FSCALC

Patti McAlister
SAS Institute Inc.

Abstract

This paper covers using the FSCALC procedure, a feature of SAS/FSP® software, in batch mode. Included is a guideline of when and how to set up a batch FSCALC job. Also covered is a description of how to set up a spreadsheet to be used for calculations and sending reports. Using the special macro variables associated with some of the screens, such as the FETCH screen, to facilitate working in a batch environment is also discussed.

When should you use FSCALC in batch?

You should consider using FSCALC in batch when you have a job that must be done repetitively over small periods of time. For instance, you would probably want to set up batch execution of FSCALC because you need to consolidate several spreadsheets every month. Any program that must be done every few days or even monthly that requires the same action every time is a prime candidate for batch processing.

Another consideration is whether what-if analysis needs to be done. Obviously if you are working in batch, you will not be able to change the values of the spreadsheet easily to do what-if analysis. In cases such as these, you are probably better off working interactively.

How do you set up FSCALC for batch execution?

To set up for batch execution you first create the spreadsheet(s) in interactive mode. FSCALC debugging is much more efficient when done interactively. It is preferrable to do the creation and testing of the spreadsheets in the interactive mode before attempting to issue the BATCH option in the PROC FSCALC statement.

Once you have the spreadsheet working interactively, run the spread-sheet in a batch job to verify there are no problems. By this time, the system should be bug free and no problems should arise. If any problems do occur, rerun the spreadsheets interactively to verify whether the problem is a global one, regardless of the environment. The most useful debugging tools are the _MESSAGE_ and PAUSE statements. These statements can be used to determine at what point in a program the difficulties arise when you are using macros. They can also be used to determine the exact value of any temporary variables that are used in the FSCALC program.

How do you use macro variables effectively in batch?

There are many macro variables available that are extremely handy for batch processing. Pages 164-166 of the SAS/FSP® User's Guide Version 5 Edition documents the macro variables for use with the new spreadsheet specification screen, the old spreadsheet specification screen, the FETCH, CLEAR, INSERT, CONSOLIDATE, and REPEAT command screens. These macro variables allow you to define values for the screens before the commands to execute the screens are pushed to the command line. When you issue a command in batch FSCALC, keep in mind that you are actually going to the screen that the command calls up. For instance, suppose you want issue the REPEAT command from the program. You are placed in the REPEAT command screen and must push an END command to leave it. While you are creating the commands to issue within a program, you should visualize which screens you must enter and exit so you can issue all of the appropriate commands.

Macro variables are handy for those occasions when you need to edit an existing spreadsheet and run a certain program with it, or if you need to issue the FETCH command as opposed to the FETCH function. As you learn more about the product you will realize the flexibility of having these screens defined within the program screen. The following example shows how to use some of these screens in the batch environment.

An Example

The following example describes how to set up an initial spreadsheet that is used as the base for the inclusion of a FETCHed data set. The data set can have any variables and be any size and the FETCH will still work successfully. After the FETCH is completed, the REPEAT command is issued to set any undefined cells in the spreadsheet to a value of 0. Calculations are then performed on the data set and a report is sent.

First, create the main spreadsheet in interactive mode. This spreadsheet is referred to as MAIN.CALC and is shown below. The cells are given an initial value of 0, but as we can see later this is not really necessary as a REPEAT will be done on any undefined cells. The cells are defined to be function numeric cells since calculations will be performed at a later date using COL2 and ROW2.

```
                                MAIN.CALC
Command ===>                                                    Define

          COL1      COL2

ROW1         0         0
ROW2         0         0
```

The program for MAIN.CALC is MAIN.PGM which is shown below.

```
                                MAIN.PGM
Command ===>


call symput('fscold2','RUN');
call symput('fscold3','PGM');
call symput('fscold4','EXE');

_command_='def; edi pgm.calc; end';
```

This program assigns a value to the macro variables beginning with the prefix FSCOLD in order to set up the old spreadsheet specification screen for the inclusion of the next spreadsheet screen. FSCOLD2 is the variable that informs FSCALC whether it should be in run mode or define mode. The next variable, FSCOLD3, indicates that the program to be run is called PGM. Finally, FSCOLD4 gives the program extension. The values for FSCOLD4 are PGM, EXE, or a blank. The old spreadsheet specification screen with the associated macro variables is given below.

```
          Old Spreadsheet Specification Screen

Spreadsheet Specification for <name of spreadsheet>

          Lock Password: &FSCOLD1   Mode (Define,Run): &FSCOLD2

           Program Name: &FSCOLD3 . &FSCOLD4


          Enter X to fetch a SAS data set: &FSCOLD5
```

Once these values are given, you can run the second spreadsheet. The reason for running another spreadsheet is to execute a FETCH command on the first spreadsheet, MAIN.CALC. The spreadsheet, MAIN.CALC, must be in define mode to execute the FETCH command because you will be adding rows and columns to the spreadsheet. You must use another spreadsheet, PGM.CALC, to push the FETCH command to the first spreadsheet because it is in define mode.

The _COMMAND_ string is defined to place the current spreadsheet, MAIN.CALC, into define mode before editing PGM.CALC, the second spreadsheet. You must visualize the screen where the EDIT command will put you. After issuing the EDIT command, you will be placed into the old spreadsheet specification screen because you are attempting to edit an existing spreadsheet, PGM.CALC. You want to enter the spreadsheet screen to execute the program, so an END statement is included in _COMMAND_ to push past the old spreadsheet specification screen and into run mode in the PGM.CALC screen.

The next action takes place in PGM.CALC. PGM.CALC now visually overlays MAIN.CALC, so that if you are watching from a terminal, MAIN.CALC cannot be seen. The JUMP command can be used to get to this screen if necessary. PGM.CALC merely exists to run the program associated with it, so that we can push FETCH into MAIN.CALC and serves no other useful purpose in this example.

```
                                PGM.CALC
Command ===>                                                  Define

          COL1

ROW1              1
```

The main work of all of the program screens is done in PGM.PGM. This program does the actual execution of the FETCH and a subsequent REPEAT to assign a value of 0 to any undefined cells. The discussion of PGM.PGM is broken into two parts so that detailed explanation of each section can be given. The entire program is displayed below.

```
                                PGM.PGM
Command ===>

/*  Part 1 of the program PGM.PGM   */

call symput('fscfet1','work');
call symput('fscfet2','temp');
call symput('fscfet9','2');
call symput('fscfet10','1');
call symput('fscfet11','ROW1');
call symput('fscfet12','COL1');
call symput('fscfet13','ROW1');
call symput('fscfet14','COL1');

_command_='jump for; fetch; end; end; jump bac; sx ';
pause;


   (continued on next screen)
```

```
   (continued from previous screen)


/*  Part 2 of the program PGM.PGM   */

call symput('fscrpt1','1');
call symput('fscrpt2','ROW1');
call symput('fscrpt3','COL1');
call symput('fscrpt4','ROW2');
call symput('fscrpt5','COL2');
call symput('fscrpt6',' ');
call symput('fscrpt7',' ');
call symput('fscrpt8','f');
call symput('fscrpt12','n');
call symput('fscrpt16','w');
call symput('fscrpt17','0');
call symput('fscrpt21','0');
call symput('fscrpt22','0');

_command_='jump for; repeat; end; jump bac; can; run main2.exe';
```

Part 1 of PGM.PGM deals directly with the execution of the FETCH command. First the macro variables associated with the FETCH screen must be set up so that FSCALC knows what is to be fetched and how it is to be placed into the spreadsheet. The FETCH screen macro variables all start with the prefix FSCFET. When reading the document you should realize that each macro variable is numbered according to its position on the FETCH screen. If you do not know what the value of the FETCH variable should be, you can get into any spreadsheet screen and issue the FETCH command. This allows you to get into the FETCH screen and determine the appropriate value for a particular macro variable. When you have finished, issue the CANCEL command to exit the FETCH screen.

The first FETCH variable given below is FSCFET1, which describes the libref of the data set to be fetched. This variable can be left blank if the libref is WORK and no previous value has been assigned to the variable. The second variable, FSCFET2, tells the actual data set to be fetched. In this case, we are fetching a data set called WORK.TEMP. The data set that is fetched in this example will always be called WORK.TEMP because these names are fixed permanently in PGM.PGM for the call symput.

FSCFET9 indicates the consolidation type. The 2 indicates that the type of consolidation is REPLACE and that any variables with the same row and column names will be replaced instead of added or subtracted. The fetch type is indicated by FSCFET10. In this instance the data set will be merged instead of inserted. Since the type of fetch is MERGE you will want to indicate the location of the merge. You actually want the data set to be brought in just after COL1, so for FSCFET11 and FSCFET13 you indicate ROW1 as being the first row and last row for the FETCH. Macro variables FSCFET12 and FSCFET14 are set to COL1. The FETCH command screen, and associated macro variables, is given below.

```
                        FETCH command screen

      Libref: &FSCFET1     Member: &FSCFET2     Password: &FSCFET3

                Enter X for variable selection: &FSCFET4
                 Enter X to transpose data set: &FSCFET5

ID Variable:              First Obs: &FSCFET7    Last Obs: &FSCFET8

  Type of Consolidation: &FSCFET9 (1=ADD,2=REPLACE,3=DIFFERENCE)
        Type of Fetch: &FSCFET10 (1=MERGE,2=INSERT)

      If Merge, Specify Block(optional)
          First Row: &FSCFET11 First Column: &FSCFET12
           Last row: &FSCFET13  Last Column: &FSCFET14

      If Insert, Specify Location(optional)
         Location: &FSCFET15  (Column name if not transposed,
                               Row name if transposed)
              Mode: &FSCFET16
```

Now the FETCH is set to run. Remember that PGM.CALC is the spread-sheet that is currently being run. In order to issue the FETCH command on the correct spreadsheet, you must first jump forward to the MAIN.CALC spreadsheet. Once you have reached the MAIN.CALC spreadsheet you are ready to issue the FETCH command. You must keep in mind the sequence of screens while you are coding the command, and issue the FETCH in the appropriate spreadsheet screen. The FETCH command screen is the first screen you enter and you will need to enter the END command to get out of this screen. The second screen associated with the FETCH command is the variable selection screen. In this instance, you are letting this screen default so that all of the variables will be brought into the spreadsheet. Issuing two END commands will bring us successfully out of the FETCH, and the data set will be brought into the screen.

Suppose that the following DATA step has been issued and that this is the data set that is being fetched into the spreadsheet.

```
DATA TEMP;
   INPUT A B C ;
   CARDS;
1 2 3
2 3 4
3 4 5
4 2 5
6 3 5
1 1 1
;
```

Since the spreadsheet has been expanded to include the columns COL1 and COL2, and the rows ROW1 and ROW2, it will be necessary to issue a REPEAT command to change any undefined cells (shown below) to the value of zero. To do this, JUMP back to the PGM.CALC spreadsheet in order to set up the REPEAT command. The final EXECUTE command pushes us past the PAUSE statement so that you can start on the next set of program statements.

```
                              MAIN.CALC
Command ===>                                                  Define

        COL1        A         B         C         COL2

ROW1    0           1         2         3         0
ROW3                2         3         4
ROW4                3         4         5
ROW5                4         2         5
ROW6                6         3         5
ROW7                1         1         1
ROW2    0                                         0
```

Part 2 of the PGM.PGM statements are preparing the REPEAT macro variables in order to issue the REPEAT command. All of the REPEAT macro variables start with the prefix FSCRPT. The first macro variable, FSCRPT1 informs FSCALC whether the repeat will take place on the cells or on the row or column characteristics. In this case we are indicating that REPEAT is to take place on cell values by typing in a 1 in the CALL SYMPUT for that variable. If we wanted to issue REPEAT on the column characteristics or row characteristics, we would use the 2 or 3 to indicate which we want to change.

The next variable FSCRPT2 indicates the first row, ROW1, of the block that is to be changed. FSCRPT3 indicates the first column, COL1, of the block. FSCRPT4 indicates the ending row of the block is ROW2. FSCRPT5 indicates the ending column is COL2. You want to change any undefined cells so the cell type, FSCRPT6, and the data type, FSCRPT7, are left blank.

FSCRPT8 and FSCRPT12 are used to indicate the data and cell type the cells are to be changed to. In this instance you are changing the undefined cells to be function numeric cells so that calculations can be done on them. FSCRPT16 indicates the format to be W with these new cell definitions, and FSCPRT17 and FSCRPT21 indicate that the decimal positions shown for these cells will be zero.

Finally, FSCRPT22 indicates the initial value that is given to these changed function numeric cells. You are assigning a value of zero to these cells so that any calculations that are done will not be influenced by these cells. The REPEAT command screen and its associated macro variables are given below.

```
                    REPEAT command screen

   Select type of repeat: &FSCRPT1 (1=CELL, 2=ROW, 3=COLUMN)

            Starting Row: &FSCRPT2     Starting Column: &FSCRPT3
              Ending Row: &FSCRPT4       Ending Column: &FSCRPT5
               Cell type: &FSCRPT6           Data type: &FSCRPT7

Please specify new cell characteristics (Cell type and Data type are
                                         required)

Cell type: &FSCRPT8       Color: &FSCRPT9          Round: &FSCRPT10
                                                    Just:  &FSCRPT11
Data type: &FSCRPT12       Attr: &FSCRPT13          Caps: &FSCRPT14
                                                    Hide: &FSCRPT15
   Format: &FSCRPT16        Dec: &FSCRPT17       Pattern: &FSCRPT18
                                                    Prot: &FSCRPT19
 Informat: &FSCRPT20        Dec: &FSCRPT21

Initial value:
   &FSCRPT22
```

Now that the macro variables for the REPEAT command are set up, you are ready to execute the commands. Again you are back in PGM.CALC so you must jump forward to get into in MAIN.CALC. After issuing the JUMP command you are ready to execute the REPEAT command. To exit the REPEAT screen, issue the END command so that the REPEAT command is completed. The spreadsheet is now ready for calculations but PGM.CALC is no longer necessary for you to continue the execution of MAIN.CALC. You will want to jump backward to PGM.CALC and once you have returned to this screen issue the CANCEL command to cancel PGM.CALC and return to MAIN.CALC. Finally, issue the RUN MAIN2.EXE command in order to associate the new program with old MAIN.CALC spreadsheet.

PGM.PGM probably can be compressed so that it is only a series of CALL SYMPUTs setting up both the FETCH and REPEAT commands, then issuing one _COMMAND_ statement, but this type of programming may be confusing. Using two _COMMAND_ statements and separating the CALL SYMPUTs makes following the code a little easier when you have to review it at a later time.

Now that the two previous program screens have been run and the REPEAT and FETCH have been accomplished, the spreadsheet will appear as follows. You are now ready to do the calculations on the spreadsheet and send your report.

```
                          MAIN.CALC
Command ===>                                                  Define

        COL1        A        B        C        COL2
ROW1       0        1        2        3        0
ROW3       0        2        3        4        0
ROW4       0        3        4        5        0
ROW5       0        4        2        5        0
ROW6       0        6        3        5        0
ROW7       0        1        1        1        0
ROW2       0        0        0        0        0
```

MAIN2.PGM is a very small program. Its main purpose is to do very simple calculations and then to issue the report. The calculations are done on COL1 and ROW2.

The _COMMAND_ variable contains the command to bring up the report screen called MAIN. Once it is up you want to SEND the report, then FREE it. Finally, issue an END statement to exit the REPORT command screen, and then issue the command CANCEL to get out of MAIN.CALC. Your report will be sent to the printer and the MAIN.CALC spreadsheet will be set to its original form. MAIN2.PGM is shown below.

```
                          MAIN2.PGM
Command ===>

col2=sum(of col1--col2);
row2=sum(of row1--row2);

_command_='report main; send; free; end; can';
```

Given below is the report screen, MAIN.REPORT, which is used to send the report.

```
                        FSCALC Report Specification
Command ===>

   Report Title: .....

Report Routing Information:              +--------------------------+
      Form Name: MAIN      (Required)    | Note:  You are in Report |
        Fileref: ........  (Optional)    |        Specification Mode |

Report Control Information:
      Convert all alphabetic text to uppercase: _
                        Ignore user pagination: _
                           Center text on page: _

Optional Report Fields:
  Global Title: _    Page No: _     Left-Justify Page Info: _
  Column Title: _       Date: X                Date Format: DATE7.
     Row Title: _       Time: X                Time Format: TIME8.


            (continued on next screen)
```

```
%LET FSCOLD2=RUN;
%LET FSCOLD3=MAIN;
%LET FSCOLD4=EXE;
PROC FSCALC C=libref.catalog.MAIN.CALC BATCH; RUN;
```

```
            (continued from previous screen)

 Title Rows: .....    .....    .....    .....    .....
 ID Columns: .....    .....    .....    .....    .....
    ID Rows: .....    .....    .....    .....    .....
  Footnotes: .....    .....    .....    .....    .....
 Scroll FORWARD To View Range Selection Screen.
 Enter CANCEL To Abort Report Generation.
 Range Selection:                    Scroll BACKWARD To View Previous
 Rows: ..................................................................
       ..................................................................
       ..................................................................
       ..................................................................
       ..................................................................
       ..................................................................

 Cols: ..................................................................
       ..................................................................
       ..................................................................
       ..................................................................
       ..................................................................
       ..................................................................
       ..................................................................
 Use Slash (/) To Indicate Page Breaks within Range Selections.
```

Keep in mind that until this point you have been doing everything mentioned above within interactive FSCALC. Once you have everything working interactively, you are ready to execute the statements in batch. MAIN.CALC has been stored in define mode, so that unless you specify the old specification screen, you will never run MAIN.EXE. You can set up the following program to run this FSCALC example in batch.

Conclusion

Running an FSCALC program in batch mode is a process that can save you time and trouble since it allows you to issue a batch job instead of running everything interactively. There are several things to consider in doing FSCALC batch jobs.

1. You should consider using FSCALC in batch when you have spreadsheets that must be run in the same manner time after time during the year.

2. When you set up an FSCALC spreadsheet for running in batch, you should first have it running smoothly in interactive mode. Debugging FSCALC program code interactively is much simpler than in batch.

3. Consider using the reserved macro variables that allow you to define screens for such commands as FETCH, CONSOLIDATE, and so on. These macro variables can be invaluable for batch execution.

References:

SAS/FSP® User's Guide, Version 5 Edition