What We Really Need is a %BY Statement

Ray Pass, ASG, Inc

Here's the scenario. You have to produce a series of different reports from the same overall population, and each report has to be run separately for each level of a specified bygroup (or combination of bygroups) in the population. The catch is that the client (boss, whatever) wants the report output to appear in bygroup order; i.e. all reports for bygroup-1, then all reports for bygroup-2, etc. Another catch is that you don't necessarily know a priori what the different levels of the bygroup are, or even how many there are; in fact, this is to be part of a routine production system where these bygroup levels will most assuredly be changing from run to run. There are undoubtedly many ways to handle a situation like this. Here is one.

```
*------------------------------------------------------------------------;
%macro repby;

   data tempby;
      set alldata(keep=byvar);
      by byvar;
      if first.byvar;

   %next:
   data tempby;
      set tempby;
      call symput('macby',byvar);
      modify tempby end=lastrec;
      call symput('lastrec',put(lastrec,1.));
      remove;
      stop;
   run;

   proc report (data=alldata(where=(byvar="&macby"));
      /*** REMAINING CODE FOR FIRST PROC ***/

   /*** INTERVENING PROCS ***/

   proc report (data=alldata(where=(byvar="&macby"));
      /*** REMAINING CODE FOR LAST PROC ***/

   %if &lastrec ne 1 &then %goto next;

   run;

%mend repby;
*------------------------------------------------------------------------;
%repby;
*------------------------------------------------------------------------;
```

The large dataset **alldata** contains all the data to be reported on. The procs used in the guts of the technique can be any procs. The method works by creating a non-duplicated list of the bygroup values (**byvar**) in a dataset (**tempby**) and then using them one at a time with a **where=** data set option to produce the reports needed. The above code assumes that dataset **alldata** is already sorted by the variable **byvar**. If it isn't, you can substitute the following code for the first **data tempby** step:

```
proc sort data=alldata(keep=byvar)
   out=tempby nodups;
run;
```

Once the list data set **tempby** is ready, its observations are used one at a time to populate the macro variable **&macby** via a **call symput** statement. As soon as each observation in **tempby** is used, it is stripped off the top of the dataset with a set of **modify** and **remove** statements. Another macro variable **&lastrec** is also created which will contain a '1' when the last observation of **tempby** is being processed. The data step processing of **tempby** is then stopped with a **stop** statement, thus freezing the values of the macro variables **&macby** and **&lastrec**.

At this point, all the report procs are run sequentially against data set **alldata**, with a **where=** data set option which only allows the observations in **alldata** with the current value of **&macby** in for processing. At the end of the report processing, a check is made to see if the last observation in **tempby** (last value of **byvar**) was just processed. This is done with a **%if** statement. If this is true (**&lastrec=1**) then the process is over. If not, the process loops back to the macro label statement **%next:,** and the next value of **byvar** is used to repopulate **&macby** for use with a new set of reports. Because the whole looping technique used here revolves around **%if-%then-%goto** processing and a *%label* statement, it must all be enclosed in a macro and cannot be used in open code. Since it is all self-contained however, there is no need for macro parameters to be used, although one could perhaps enhance the technique even further with their use.

The author of this paper can be contacted as follows:

Ray Pass
ASG, Inc.
1100 Summer Street
Stamford, CT 06905

Voice: (203) 356-9540
FAX: (203) 967-8644
EMS: raypass@worldnet.att.net