Rapid Record Matching via FORMATS and CNTLIN Datasets
Mike Zdeb, New York State Department of Health

INTRODUCTION
At some point, you might be confronted by one or more of the following tasks: selecting specific records from a file of raw data or a SAS dataset based on the value of one of the variables in the data; combining information from different files based on the value of one the variables common to different files; updating the information in a SAS dataset. Different techniques come to mind, including data step merges and/or updates, and possibly PROC SQL. There is another procedure that may not come to mind as readily, but can also get the job done (often in less time than other methods) - PROC FORMAT. The following examples will demonstrate how to use PROC FORMAT in lieu of merge, update, or PROC SQL.

EXAMPLE 1/SELECTING RECORDS - You have a list of ID numbers and must extract records with the given IDs from a file of raw data. If the list of ID numbers is short, you can enter the format manually:

```sas
proc format;
value $idok
  'A23456','A54321','Z34567',
  'Q65432','U56982' = 'OK' other='NO';
run;

data mike;
  infile mydata;
  input @01 id $char6.
  @07 (etc.)
  ;
  if put(id,$idok.) eq 'OK';
run;
```

What if the number of records to be extracted was large, e.g., you are given a file that contains 500 ID numbers and you want to extract a record for each ID from a larger file? You definitely do not want to retype all the IDs in PROC FORMAT. Why not let SAS do all the work and create the format for you via a CNTLIN dataset.

If your file of ID numbers that looks like this...

A12345
B23456

...then...

```sas
Z76543
e tc. (through the 500th record)
you can create a CNTLIN dataset that can be used by PROC FORMAT.

data select
  (keep=fmtname start label hlo);
  retain fmtname '$idok' label 'OK';
  infile myidnums end=last;
  input start $char6.;
  output;
  if last then do;
    hlo='o';
    label='NO';
    output;
    stop;
  end;
run;

data mike;
  infile mydata;
  input @01 id $char6.
  @07 (etc.)
  ;
  if put(id,$idok.) eq 'OK';
run;
```

When creating the CNTLIN dataset SELECT, all records with good ID numbers will receive a formatted value of 'OK', while all others will receive a formatted value 'NO' as a result of using the HLO variable in the datastep - using a value of 'o' for the HLO variable is the same as using an OTHER on the left hand side of the =s in writing your own PROC FORMAT.

What you may have done in this situation is: read all the IDs into a SAS data set; read all the observations from the large file into a SAS data set; sort both data sets by ID; merge the two data sets by ID and retain those observations where the data sets matched ID numbers, or match the datasets using PROC SQL. The PROC FORMAT approach eliminates a lot of steps (and CPU time).

EXAMPLE 2/MATCHING RECORDS - You have two files. The first (a master file) contains an social security number, name, and address. The second contains a social security number plus some additional demographic data. You want to add
information to the master file, retaining only matched records.

data demog (keep=age etc.)
socsec (keep=fmtnme start label hlo);
retain fmtnme "$ssok";
infile filedem;
* start is the social security number;
input
@01 start $char9.
@010 age 3.
@013 (etc. - more demographic data);
label = put(_n_,6.);
if last then do;
   hlo='0';
   label='000000';
   output socsec;
end;
run;

proc format cntlin=socsec;
run;

data first (keep=sssec name address);
infile first end=last;
input
@01 sssec $char9.
@010 name $char20.
@030 address $char30.
@nrec = input(put(sssec,$ssok,),6.);
if nrec ne 0 then do;
   set demog point=nrec;
   output;
end;
run;

There are several features you should have noticed that differentiate this example from the previous example of record selection. In this example, the DATA step that creates the CNTLIN dataset SOCSEC also creates a data set, DEMOG. Instead of writing a '1' to the external file as the formatted value (i.e., the entry on the right side of the '=' in the PROC FORMAT), a record number (_N_) is written. Once again, an HLO variable is used to take care of the situation where a social security number is not present in both the master file and the file of demographic data. As the master file is read, the social security numbers are checked to see if they existed in the demographic dataset. Those that did result in a matched set of data being written to the dataset BOTH.

Once again, another approach in this situation would be to merge the data sets after reading and sorting both data sets, or to match them using PROC SQL. The method shown in this example eliminates the sorts and creates matches 'on the fly' as the second file is read (once again, less CPU time).

**EXAMPLE 3/UPDATING A DATASET** - In the previous example, records were matched based on the value of social security number and only matched records were kept. If the task is changed, i.e., we are given a list of social security numbers and updated addresses for some records and must update selected records on the master file, PROC FORMAT can still be used. The SAS code must be modified slightly...

data newaddr (keep=address)
socsec (keep=fmtnme start label hlo);
retain fmtnme "$ssok";
infile filedem;
* start is the social security number;
input
@001 start $char9.
@010 address $char30.
label = put(_n_,6.);
if last then do;
   hlo='0';
   label='000000';
   output socsec;
end;
run;

proc format cntlin=socsec;
run;

Data master;
* MASTER is a SAS dataset, one variable is social security number;
set master;
nrec=input(put(sssec,$ssok,),6.);
if nrec ne 0 then set newaddr point=nrec;
run;

There are a number of different approaches to updating the values of variables in a SAS dataset (that's one of the great features of SAS). In lieu of PROC FORMAT, you might index the master file on social security number and use the index in updating observations. You could also use an UPDATE statement in a data step. PROC FORMAT is just another alternative.