Paper 971
INSTANT FORMATS in a BLINK with PROC FORMAT CNTLIN=

Christopher J. Boniface, U.S. Census Bureau;

ABSTRACT

Do you need to create a format instantly? Does the format have a lot of labels and it would take a long time to type all the codes and labels by hand? Sometimes a SAS programmer needs to create a user-defined format for hundreds or thousands of codes. There must be an easy way to accomplish this without having to type all of the codes. SAS does provide a way to complete a user-defined format without having to type any codes. If the codes and labels are in a text file, SAS dataset, Excel file, or in any file that can be converted to a SAS dataset, then a SAS user-defined format can be created on the fly. The CNTLIN=option of PROC FORMAT allows a user to create a user-defined format or informat from raw data or a SAS file. This paper will demonstrate a couple of quick tips to show how to create two user-defined format's instantly from a raw text file on our Census Bureau website. It will further demonstrate how to use these user-defined formats for the final report and for the final output data set from PROC TABULATE. The paper will focus on the CNTLIN= option of PROC FORMAT and not the CNTLOUT= option.

INTRODUCTION

The U.S. Census Bureau's 2010 Census Special Tabulation Program provides data users with the option to have user-defined tabulations created from decennial census microdata on a cost-reimbursable basis. When requesting a special tabulation, the sponsor should provide a preliminary, general specification of the data needed. We will ask them some specific questions, and then work with them to develop a final, detailed specification that documents their data needs and geographic requirements. For additional information on the U.S. Census Bureau's Special Tabulation Program, see http://www.census.gov/population/www/cen2010/spec-tab/.

Typically, a sponsor may ask to see cross tabulations by various U.S. geography levels. There are many different Census geography levels, such as state, county, tract, and place. While there are user-defined formats for some of the more common requests, sometimes there is not a user-defined format for certain geographic levels. Let's consider a hypothetical special tabulation. A sponsor requests a special tabulation by United States place for Florida. The Census Bureau defines a place as a concentration of population; a place may or may not have legally prescribed limits, powers, or functions. This concentration of population must have a name, be locally recognized, and not be part of any other place. There are 920 places in Florida. If there isn't a pre-existing user-defined format for place, a decision has to be made. The programmer has to decide whether to type all of the place names by hand with their corresponding place codes or to find an easier way to create a dynamic user-defined format quickly. This paper will demonstrate how to create a user-defined format quickly from a raw text file on our Census Bureau website.

Sample Case Study

A fictitious sponsor has requested a decennial special tabulation from the Census Bureau. They want to see counts of people by race, age, and sex by place for Florida. They indicate that they want to see the names of each place in the TABULATE report and additionally, they want a final data set with both the place names and place codes in it. The sample case study will show how to get the place names in the report by creating a SAS user-defined format using the CNTLIN= option of PROC FORMAT. However, after running a PROC TABULATE, not all of the places in our study have crossings by age, sex, and race. Some crossings by place are missing. The sponsor wants to see all places in the final output even if there are no values for a particular place cross tabulation. The PRELOADFMT option in PROC TABULATE together with our user-defined format will ensure that the places with missing cross tabs are displayed in our final report. Lastly, the sponsor requests that they receive a final output dataset from TABULATE with both the place codes and place names in it. In order to get both the place name and the place code in the final output, a second user-defined format needs to be created with the CNTLIN= option. This will in effect create a reverse format (example- place name=place code). The paper will show how easy it is to create user-defined formats

quickly using PROC FORMAT with the CNTLIN= option. It will also show how to use user-defined formats effectively in the final PROC TABULATE report and in our final output data set.

Finding the Place Names

If you don't have a place user-defined format handy anywhere, you need to decide how to create one. In our case study, there is just one state, Florida. Fortunately, this fictitious special tabulation is only for Florida and not for all states. If that were the case, there would be thousands of places. For Florida, there are 920 places. While it would be possible to type in the place names and codes for 920 places, it would still be better not to have to type any of the labels and codes. The next question is whether the names and codes for the places in Florida are online or in a raw text file somewhere. Fortunately, the Census Bureau website has many standardized geography codes available. The American National Standards Institute codes (ANSI codes) are standardized numeric or alphabetic codes issued by the American National Standards Institute (ANSI) to ensure uniform identification of geographic entities through all federal government agencies. The following Census Bureau website is the starting point: http://www.census.gov/geo/reference/ansi.html. The link has various lists of Census Bureau geography files by state. First, you need to click on the link for "Places" on the site. A couple of clicks later, you find the list for the places for the state of Florida. This is the list of places by place code. Now, you simply need to convert this raw file on our website into a raw text file and then into a SAS dataset. Output 1 below shows a partial listing of the raw text file on the website containing the Florida place codes and names of interest. Note: for the purposes of brevity for this paper, I will show just four particular places: Cocoa Beach city, Daytona Beach city, Orlando city, and Tampa city.

FL|12|13175|Cocoa Beach city|Incorporated Place|A|Brevard County
FL|12|16525|Daytona Beach city|Incorporated Place|A|Volusia County
FL|12|53000|Orlando city|Incorporated Place|A|Orange County
FL|12|71000|Tampa city|Incorporated Place|A|Hillsborough County

Output 1. Raw text file listing (partial) of Place codes and names for FL on Census website

Converting the Raw Text Place file

By clicking on "File -> Save as" from the pull-down menu on the Census website you can save the file in Output 1 as a raw text file (or .txt file) quickly. Now, you need to convert the raw text file into a SAS data set using simple data step code. The code below shows how to read the text file into a SAS dataset.

```
/* Reads in the Place codes and Place Names for Florida */

libname inp '/places/inp';
 data inp.fl_place;
  length state tabst $2 place_code $5 place type cou_name $30 fstatus $1;
  filename place '/place/inp/st12_fl_places.txt' ;
  infile place dlm='|';
  input state $ tabst $ place_code $ place $ type $ fstatus $ cou_name ;
 run;
```

Input Control Data Set

Now that the place information is in a SAS data set, you need to convert the data set into a user-defined format. The first thing to do is to convert the above SAS dataset into an input control data set. The FORMAT procedure uses the data in the input control data set to construct informats and formats. Thus, you can create informats and formats without writing INVALUE, PICTURE, or VALUE statements. According to SAS, the input control data set must have these characteristics:

For both numeric and character formats, the data set must contain the variables FMTNAME, START, and LABEL, which are described below. There are additional variables that could be specified, but these additional variables are not always required. The three required variables will suffice for our problem.

FMTNAME - specifies a character variable whose value is the format or informat name.
START - specifies a character variable that gives the range's starting value.
LABEL - specifies a character variable whose value is associated with a format or an informat.

If you are creating a character format or informat, then you must either begin the format or informat name with a dollar sign ($) or specify a TYPE variable with the value **C**. For our case study two formats need to be created, one with the actual place names that will be used in our TABULATE report, and a second reverse format after the TABULATE to retrieve the place codes back again. In Program 1 below, the place_code variable contains the place codes, such as 13175. The place variable contains the place names. The place_code variable will be our "Start" variable in the control input data set. Thus, rename place_code to Start. Similarly, the place variable will be our "Label" variable. Thus, rename it to Label. So in terms of creating a user-defined format with PROC format, you can think of this as 'start'='label' or '13175'='Cocoa Beach city'. Finally, assign the value, $placfmt, to the fmtname variable in order to name the format. Notice that since this is a character format, a dollar sign ($) and a TYPE variable with the value 'C' are specified. Lastly, create the first input control data set (inp.place_fmt). This data set will be used to create the user-defined format, named $placfmt., in order to get the place names. This format, $placfmt., will be used in our PROC TABULATE with PRELOADFMAT later to ensure that all places appear in the table. Table 1 shows the input control data set for format, $placfmt.

```
/* Program 1. Create format with Place Names/Labels for each Place code */
data inp.place_fmt(keep=start label fmtname);
 set inp.fl_place;
  if place_code in('13175','16525','53000','71000');
 retain fmtname '$placfmt' type 'C';
RENAME place_code = Start;
RENAME place = Label;
run;
```

Place Format

Start	Label	fmtname
13175	Cocoa Beach city	$placfmt
16525	Daytona Beach city	$placfmt
53000	Orlando city	$placfmt
71000	Tampa city	$placfmt

Table 1. Format 1 $placfmt. Place Code = Place Name

In Program 2 below, create a second user-defined format, named $plac2fmt. This is the reverse format in the form Place name=Place code. This format will be used in a data step after our PROC TABULATE in order to get the place codes back from the final TABULATE output dataset. In this case, specify the variable place as the "Start" variable and place_code as the "Label" variable. Thus, for Cocoa Beach city our range would look like 'Cocoa Beach city' = '13175'. Lastly, create the second input control data set (inp.plac2_fmt), which will be used to create the $plac2fmt. Table 2 shows the input control data set for format $plac2fmt.

```
/* Program 2. Create a 2nd reverse format to get Place codes for each Place name */

data inp.place2_fmt(keep=start label fmtname);
 set inp.fl_place;
  if place_code in('13175','16525','53000','71000');
 retain fmtname '$plac2fmt' type 'C';
RENAME place = Start;
RENAME place_code = Label;
run;
```

Reverse Place Format

Label	Start	fmtname
13175	Cocoa Beach city	$plac2fmt
16525	Daytona Beach city	$plac2fmt
53000	Orlando city	$plac2fmt
71000	Tampa city	$plac2fmt

Table 2. Format 2 $plac2fmt. Place Name = Place Code

PROC FORMAT with CNTLIN= option

Now that you've created two input control data sets, you need to read them in using PROC FORMAT with the CNTLIN= option. You specify an input control data set with the CNTLIN= option in the PROC FORMAT statement. The FORMAT procedure uses the data in the input control data set to construct informats and formats.

```
/*Read in place_fmt which contains $placfmt. Form is '13175'='Cocoa Beach city' */
 proc format CNTLIN=inp.place_fmt;
 run;

/*Read in place_fmt2 which contains $plac2fmt. Form is 'Cocoa Beach city'='13175'
 proc format CNTLIN=inp.place2_fmt;
 run;
```

The two formats are now available for our special tabulation. Below is the PROC TABULATE program to create the final report. Notice, the PRELOADFMT option is not used on the class statement for the place_code variable.

```
title1 'Final Report with Missing Places';
proc tabulate data=recodes out=sums(drop=_type_ _page_ _table_);
 class age / order=formatted preloadfmt mlf missing;
 class imprace sex/ order=formatted missing;
 class place_code / order=formatted missing;
 var count;
 table place_code='PLACE Name'*race,age=''*sex='' * (count=''*sum='') / printmiss;
 format age agef. place_code $placfmt.;  run;
```

Table 3 below shows the results. Notice that one place, Orlando city, is missing in the report. There were no observations in the data for this particular report for Orlando city. In order to get Orlando city in the report, specify the PRELOADFMT option on the class statement for the place_code variable.

Final Report with Missing Places

		11-13		14 - 16	
		F	M	F	M
PLACE Name	Race				
Cocoa Beach city	1	2	1	0	1
	2	1	0	2	1
Daytona Beach city	1	1	0	1	2
	2	1	4	1	1
Tampa city	1	0	1	0	2
	2	1	0	1	0

Table 3. Missing Place - Orlando city not in table

[1] All data and numbers in the tables of this paper in the Sample Case Study are fictitious.

In order to show all place names in the table, simply add the PRELOADFMT option to the class statement for the place_code variable (see code below). Note the use of options missing='0' to convert missing values to zero.

```
options missing=0;
proc tabulate data=recodes out=sums(drop=_type_ _page_ _table_);
 class age / order=formatted preloadfmt mlf missing;
 class imprace sex/ order=formatted missing;
 class place_code / order=formatted preloadfmt missing;
 var count;
 table place_code='PLACE Name'*race,age=''*sex='' * (count=''*sum='') / printmiss;
 format age agef. place_code $placfmt.;   run;
```

Table 4 below shows the results. Notice that Orlando city now appears in the table because the PRELOADFMT option tells SAS to use all of the values for the place_code variable contained in the $placfmt. format, regardless of whether there is any data for that value in the data set.

Final Report with All PLACES

		11-13		14 - 16	
		F	M	F	M
PLACE Name	**Race**				
Cocoa Beach city	**1**	2	1	0	1
	2	1	0	2	1
Daytona Beach city	**1**	1	0	1	2
	2	1	4	1	1
Orlando city	**1**	0	0	0	0
	2	0	0	0	0
Tampa city	**1**	0	1	0	2
	2	1	0	1	0

Table 4. All Places including Orlando city

Lastly, the sponsor indicated that they want a final output data set from PROC TABULATE in addition to the above report. Moreover, they want to see both the place names and place codes in the final data set. However, a quick look at the output data set (work.sums) from PROC TABULATE shows that the dataset has the place Names, but not the place codes. Table 5 below shows the output sums dataset (partial output for display) from PROC TABULATE.

TABULATE dataset - No PLACE Codes

PLACE Name	Age	Race	Sex	Count
Cocoa Beach city	11-13	01	F	2
Daytona Beach city	11-13	01	F	1
Orlando city	11-13	01	F	0
Tampa city	11-13	01	F	0
Cocoa Beach city	11-13	02	F	1
Daytona Beach city	11-13	02	F	1
Orlando city	11-13	02	F	0
Tampa city	11-13	02	F	1

Table 5. Output TABULATE dataset SUMS - No Place Codes

In order to create a variable to get the place codes in addition to the place names in the final data set, an extra data step needs to be to run after PROC TABULATE. That is accomplished by using the second user-defined format ($plac2fmt.) with the put function in a data step. The program below shows the post-processing in order to get the place codes in addition to the place names in our final data set.

```
/* Get Place code with $plac2fmt reverse format i.e. - 'Orlando city' = '53000' */
   data final (rename=(count_Sum=count place_code=place_name ));
    set sums(obs=8);
    new_place_code=put(place_code,$plac2fmt.); /* Get PLACE code back */ run;
```

Table 6 below shows the final output dataset (partial listing for display) with both the Place names and codes in it.

Final Dataset with PLACE Names & PLACE Codes

PLACE Name	PLACE Code	Age	Race	Sex	Count
Cocoa Beach city	13175	11-13	01	F	2
Daytona Beach city	16525	11-13	01	F	1
Orlando city	53000	11-13	01	F	0
Tampa city	71000	11-13	01	F	0
Cocoa Beach city	13175	11-13	02	F	1
Daytona Beach city	16525	11-13	02	F	1
Orlando city	53000	11-13	02	F	0
Tampa city	71000	11-13	02	F	1

Table 6. Final output data set with both Place Names and Place Codes

CONCLUSION

It is easy to create formats instantly on the fly using PROC FORMAT with the CNTLIN= option. There is no need to type codes and labels out by hand, especially if you have lots of them. As long as the codes and labels are in a raw file in some format, SAS can read them into an input control data set to be used as a parameter in the CNTLIN= option. SAS will create a user-defined format automatically from the input control data set. This paper demonstrates how to create and use two user defined formats using the CNTLIN= option of PROC FORMAT. Moreover, it shows how to use the two user-defined formats. First, I showed how to use a user-defined format with PROC TABULATE and PRELOADFMT to get all the missing Place names in our final report. Secondly, I showed how to use a second user-defined format to get the Place codes in our final output data set. It's easy to create instant formats in a blink.

REFERENCES

BASE SAS® 9.2 Procedures Guide

American Standards National Institute (ANSI) Codes http://www.census.gov/geo/reference/ansi.html.

Your comments and questions are valued and encouraged. Contact the author at:

Name: Christopher J. Boniface
U.S. Census Bureau
Washington D.C. 20233
Work Phone: (301)763-5769
E-mail: christopher.j.boniface@census.gov