Processing multiple formats with input and output control data sets in PROC FORMAT

Thomas E. Billings, Kaiser Permanente, Oakland, California

Abstract

The CNTLIN= option of PROC FORMAT lets you create a format from an input control SAS® data set. Similarly, the CNTLOUT= option lets you create an output control SAS data set that reflects the format. Both the CNTLIN= and CNTLOUT= options can handle multiple formats, in a single invocation. However, this must be done with care as the way the SAS system handles these formats/files, might not be what you expect. This paper gives examples and tips for users to successfully handle multiple formats with input/output control data sets in PROC FORMAT.

Introduction: using input and output control data sets for single formats

PROC FORMAT allows you to create a format (or informat) from a “special” SAS data set -- an input control data set -- via the CNTLIN= option. Similarly, the CNTLOUT= option lets you create a special data set -- an output control data set -- that describes an existing format or informat. Input and output control data sets can be very useful at times, and serious SAS programmers are advised to develop skill in handling these types of data sets.

To understand input control data sets, first think of a typical invocation of PROC FORMAT to create, say, a very simple format. It would include a VALUE statement with the format name, and then a set of assignment statements specifying:

- single_input_value => resultant_value
- input_range#1 to input_range#2 => resultant_value
- multiple_input_values => resultant_value.

The logic specified in the VALUE statements is used in a table lookup process to yield the resultant output value for the format invocation. The input control data set contains the lookup table defined in the PROC FORMAT invocation, where each logical assignment statement requires at least one row/case in the CNTLIN= data set. Each row of the input control data set includes the variables:

- START: single_input_value or input_range#1 or one value from multiple_input_values
- END: optional -- defaults to value of START; input_range#2
- LABEL: specifies resultant_value
- TYPE: specifies format/informat type, e.g. numeric, character, etc.
- FMTNAME: specifies format name.

Additional special variables, e.g., FUZZ, HLO, SEXCL and EEXCL, must be defined and provided in some cases. Output control data sets have a structure similar to input control data sets.

The SAS system documentation on input and output control data sets is rather minimal, and leaves out a number of useful points. This paper serves to illustrate a few of the more interesting points.

Variable types for START, END

Let’s use a few simple formats to investigate the behavior of PROC FORMAT. The formats will be based on a common grading scheme that one might see in a school. Let’s start with the raw data:
options nocenter;

data rawdata4formats;
  infile datalines dlm=',' dsd;
  length lettergrade $ 1 gradedesc $ 20;
  input lowend highend lettergrade gradedesc;
  datalines;
  ...snipped... see printout below
  ;
  run;

proc print data=rawdata4formats;
  var lowend highend lettergrade gradedesc;
  title "Raw data for format experiments";
  run;

which yields:

<table>
<thead>
<tr>
<th>Obs</th>
<th>lowend</th>
<th>highend</th>
<th>lettergrade</th>
<th>gradedesc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90</td>
<td>100</td>
<td>A</td>
<td>Excellent</td>
</tr>
<tr>
<td>2</td>
<td>80</td>
<td>89</td>
<td>B</td>
<td>Good</td>
</tr>
<tr>
<td>3</td>
<td>70</td>
<td>79</td>
<td>C</td>
<td>Average</td>
</tr>
<tr>
<td>4</td>
<td>55</td>
<td>69</td>
<td>D</td>
<td>Fair/low pass</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>54</td>
<td>F</td>
<td>Fail/no credit</td>
</tr>
<tr>
<td>6</td>
<td>.</td>
<td>.</td>
<td>X</td>
<td>No grade</td>
</tr>
</tbody>
</table>

In this example, we have a grade (A-D,F,X), a numeric range for each value (missing for X), and a description of the grade. (This is a simplified example; we adopt the convention that grades are always integer-valued, and between 0 and 100.)

Let's start by making a format that maps numeric grade to letter grade, using the raw data file created above:

    /* create input control data set for numeric format */

data num2letter;
  set rawdata4formats;
  start = lowend;
  end = highend;
  label = lettergrade;
  fmtname = 'num2letter';
  type = 'n';
  if (missing(start)) then hlo = 'o';
  keep start end label fmtname type hlo;
  output;
  run;

    /* use input control data set to create format; also get output control data set; check output file */

proc format /*lib=work */ cntlin=num2letter cntlout=num2letter_cds;
  /* select num2letter; * statement not needed here; */
  run;

proc contents data=num2letter_cds;
  run;

proc print data=num2letter_cds;
  run;
Note that START, END variables are numeric in the CNTLIN= data set. However the PROC CONTENTS for the CNTLOUT= data shows that these variables are character in the (derived) output control data set:

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>END</td>
<td>Char</td>
<td>16</td>
<td>Ending value for format</td>
</tr>
<tr>
<td>2</td>
<td>START</td>
<td>Char</td>
<td>16</td>
<td>Starting value for format</td>
</tr>
</tbody>
</table>

And the PROC PRINT shows us a typical CNTLOUT= data set for a numeric format:

```
D L
F                                               D                            D A A
M                                               E L        P     N         D I T N
T S                          L      F E        R     O   S E   E G A G
N T                          A      A N    F   E M F E T E E   C 3 T U
0 A A E BM MUG U F UIDY X X H S S Y A
b M R N E I ALT Z I LI I P C C L E E P G
s E T D L N X T H Z X T L T E L L O P P E E
1 NUM2LETTER                0               54 F 1 4 0 1 1 E-12 0 0 N N N
2 NUM2LETTER               55              69 D 1 4 0 1 1 E-12 0 0 N N N
3 NUM2LETTER              70               79 C 1 4 0 1 1 E-12 0 0 N N N
4 NUM2LETTER              80               89 B 1 4 0 1 1 E-12 0 0 N N N
5 NUM2LETTER             90              100 A 1 4 0 1 1 E-12 0 0 N N N
6 NUM2LETTER **OTHER**    **OTHER**        X 1 4 0 1 1 E-12 0 0 N N N O
```

This illustrates the tip:

T1. START, END may be numeric in a CNTLIN= data set (for numeric formats), but are always character variables in CNTLOUT= data sets.

Can START, END be character variables for a numeric format in a CNTLIN= data set? The answer is yes. The code:

```plaintext
data num3letter;
set rawdata4formats;
length start end $3;
start = put(lowend,3.);  ***discussed below;
end = put(highend,3.);
label = lettergrade;
fmtname = 'num3letter';
type = 'n';
if (missing(lowend)) then hlo = 'o';
keep start end label fmtname type hlo;
output;
run;
```

produces a printout similar to the one above, with the only notable difference being FMTNAME.

Note the subtle difference above in the code used to identify missing grades, to set the (other=) default value (hlo = 'o'). In the num2letter example above "missing(start)" is used, where start is numeric. In num3letter, "missing(lowend)" is used rather than "missing(start)". Savvy readers know that the missing function can work with character or numeric variables/expressions, so why not "missing(start)" in the num3letter example as well? This is because the line marked *** above, does not produce a missing character value when the numeric variable lowend is missing. (Note: in num3letter, a missing character value is produced if you set OPTIONS MISSING = '.') We have not set the preceding option, so we test the variable lowend rather than variable start in num3letter. This can be seen in the PROC PRINT for num3letter, the raw CNTLIN= file:
This illustrates the tip:

T2. **START, END** can be character variables for a numeric format, in CNTLIN= files.

### Working with multiple formats in control data sets

Recognizing that a variable cannot be simultaneously both character and numeric in a SAS data set, we deduce the next tip from the above discussion.

T3. When creating both numeric and character formats in a single input control data set, **START and END must be character**, i.e., numeric range values must be converted to character.

This raises the interesting question: when converting numbers to character for the **START/END variables**, using the PUT function, what SAS DATA step language numeric formats give results that are readable by PROC FORMAT? This question is not answered in this paper.

We can also deduce the tip:

T4. When creating multiple formats in a single input control data set, the length of **START, END, LABEL variables** must be adequate to support the largest values needed over the entire set of formats contained in the control data set.

Let's now try to make two formats in a single invocation of PROC FORMAT:

```sas
* two formats:
  grade description -> numeric range (as character string)
  letter grade -> numeric range (as character string);
data twoatonce;
set rawdata4formats;
start=gradedesc;
end=gradedesc; * optional - defaults to same value as start;
if (not(missing(lowend))) then labelpart1 = put(lowend, 3.);
else labelpart1 = 'N/A';
if (not(missing(highend))) then labelpart2 = put(highend, 3.);
else labelpart2 = 'N/A';
label = cat(labelpart1, ' - ', labelpart2);
type = 'c';
fmtname = 'grade2range';
output;
start = lettergrade;
end = lettergrade;
label = gradedesc;
* type = 'c'; * redundant;
fmtname = 'letter2desc';
output;
keep start end label fmtname type;
run;
```

```
proc print data=twoatonce;
run;
```

The preceding code pulls in one row from rawdata4formats, and creates two rows in an input control data set, one for each format. The resultant data set looks like (edited):

<table>
<thead>
<tr>
<th>Obs</th>
<th>start</th>
<th>end</th>
<th>label</th>
<th>fmtname</th>
<th>type</th>
<th>hlo</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>.</td>
<td>.</td>
<td>X</td>
<td>num3letter</td>
<td>n</td>
<td>o</td>
</tr>
</tbody>
</table>

...snipped...
Obs    start             end               label             type      fmtname
1    Excellent         Excellent          90 - 100          c      grade2range
2    A                 A                 Excellent          c      letter2desc
3    Good              Good               80 - 89          c      grade2range
4    B                 B                 Good               c      letter2desc

Note: total of 12 rows in data set

Now, let’s try making 2 formats at once using CNTLIN=, and check the results via CNTLOUT=:

    proc format /*lib=work */ cntlin=twoatonce;
    run;

    proc format /*lib=work */ cntlout=twoatonce_cds;
    run;

    proc print data=twoatonce_cds;
    run;

with the resulting output control data set:

    Obs   FMTNAME        START             END        LABEL     MIN MAX DEFAULT LENGTH  FUZZ
    1 NUM2LETTER                 0               54 F          1   40    1       1    1E-12
    2 NUM2LETTER                55               69 D          1   40    1       1    1E-12
    3 NUM2LETTER                70               79 C          1   40    1       1    1E-12
    4 NUM2LETTER                80               89 B          1   40    1       1    1E-12
    5 NUM2LETTER                90              100 A          1   40    1       1    1E-12
    6 NUM2LETTER  **OTHER**        **OTHER**        X          1   40    1       1    1E-12
    7 NUM3LETTER                 0               54 F          1   40    1       1    1E-12
    8 NUM3LETTER                55               69 D          1   40    1       1    1E-12
    9 NUM3LETTER                70               79 C          1   40    1       1    1E-12
   10 NUM3LETTER                80               89 B          1   40    1       1    1E-12
   11 NUM3LETTER                90              100 A          1   40    1       1    1E-12
   12 NUM3LETTER  **OTHER**        **OTHER**        X          1   40    1       1    1E-12
   13 GRADE2RANGE No grade         No grade         N/A - N/A  1   40    9       9        0
   14 LETTER2DESC X                X                No grade   1   40    8       8        0

Oops! That is not what we expected. Comments:

1. We left off a SELECT statement, so we got all the formats in the work library, in the CNTLOUT= data set.

2. Problem: GRADE2RANGE and LETTERDESC should have 6 rows each, not 1.

The SASLOG shows us that something went wrong with the 2 target formats:

    96   proc format /*lib=work */ cntlin=twoatonce;
    NOTE: Format $GRADE2RANGE has been output.
    NOTE: Format $LETTER2DESC has been output.
    NOTE: Format $GRADE2RANGE is already on the library.
    NOTE: Format $LETTER2DESC has been output.
    NOTE: Format $GRADE2RANGE is already on the library.
    NOTE: Format $LETTER2DESC has been output.
    NOTE: Format $GRADE2RANGE is already on the library.
    NOTE: Format $LETTER2DESC has been output.
    NOTE: Format $GRADE2RANGE is already on the library.
    NOTE: Format $LETTER2DESC has been output.
    NOTE: Format $GRADE2RANGE is already on the library.
    NOTE: Format $LETTER2DESC has been output.
    NOTE: Format $GRADE2RANGE is already on the library.
    NOTE: Format $LETTER2DESC has been output.
    NOTE: Format $GRADE2RANGE is already on the library.
    NOTE: Format $LETTER2DESC has been output.
    NOTE: Format $GRADE2RANGE is already on the library.
    NOTE: Format $LETTER2DESC has been output.
    NOTE: Format $GRADE2RANGE is already on the library.
    NOTE: Format $LETTER2DESC has been output.
    NOTE: Format $GRADE2RANGE is already on the library.
    NOTE: Format $LETTER2DESC is already on the library.


NOTE: Format $LETTER2DESC has been output.
NOTE: Format $GRADE2RANGE is already on the library.
NOTE: Format $GRADE2RANGE has been output.
NOTE: Format $LETTER2DESC is already on the library.
NOTE: Format $LETTER2DESC has been output.
97 run;

NOTE: PROCEDURE FORMAT used (Total process time):
    real time           0.00 seconds
    cpu time            0.00 seconds

NOTE: There were 12 observations read from the data set WORK.TWOATONCE.

We now have tips T5, T6, and T7 is demonstrated below.

T5. If a SELECT statement is not used when using CNTLOUT=, then all formats in the LIB= or work default catalog will appear in the output control data set. If that is not what you want, use a SELECT statement. A similar point applies to CNTLIN= data sets.

And we deduce the following tip as well:

T6. When using SELECT in an invocation of PROC FORMAT that has both CNTLIN= and CNTLOUT= options, the (same) SELECT applies to both input and output control data sets. If you want different SELECT sets for CNTLIN= vs. CNTLOUT= options, use two invocations of PROC FORMAT, one with (only) CNTLIN=, and one with (only) CNTLOUT=. (Alternately, use WHERE= option testing FMTNAME to specify target formats for CNTLIN=, CNTLOUT= data sets.)

Hence if you are working with multiple formats in control data sets, you may find use of SELECT to be a good idea, and/or you may find it necessary to use 2 invocations of PROC FORMAT on occasion.

T7. When a CNTLIN= data set contains multiple formats, the rows must be grouped for each individual format.

Let’s try and see if grouping them will fix the problem of lost rows:

```sql
proc sort data=twoatonce;
    by fmtname;
run;

proc format /*lib=work */ cntlin=twoatonce cntlout=twoatonce_cds;
    select $grade2range $letter2desc;
run;

proc print data=twoatonce_cds;
run;
```

which yields a data set that looks like:

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>M</th>
<th>T</th>
<th>S</th>
<th>L</th>
<th>N</th>
<th>T</th>
<th>A</th>
<th>0</th>
<th>A</th>
<th>E</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GRADE2RANGE</td>
<td>Average</td>
<td>Average</td>
<td>70 – 79</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>GRADE2RANGE</td>
<td>Excellent</td>
<td>Excellent</td>
<td>90 – 100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>GRADE2RANGE</td>
<td>Fail/no credit</td>
<td>Fail/no credit</td>
<td>0 – 54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>GRADE2RANGE</td>
<td>Fair/low pass</td>
<td>Fair/low pass</td>
<td>55 – 69</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>GRADE2RANGE</td>
<td>Good</td>
<td>Good</td>
<td>80 – 89</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>GRADE2RANGE</td>
<td>No grade</td>
<td>No grade</td>
<td>N/A – N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The CNTLOUT= data set above shows that the formats were recreated correctly. The SASLOG also looks normal here.

Note that the CNTLIN= requirement appears to be:

- raw data for formats are grouped by format name, but
- do not have to be "sorted by" format name.

PROC SORT is used above to group the rows, for convenience. CNTLOUT= data sets are not sorted, but they are grouped (see PROC CONTENTS output below for confirmation).

**PROC FORMAT statement options of interest**

T8. PROC FORMAT statement: you may need to use these options with CNTLOUT=, to manage label and START/END variable length: MAXLABELN=, MAXSELEN=. These options do not apply to CNTLIN= data sets; for them you have some length control via the variables: MAX, MIN, LENGTH, and via the LENGTH or ATTRIB statements for the START, END, LABEL variables.

T9. PROC FORMAT statement: NOREPLACE is another option that is useful at times with CNTLIN= data sets.

**Contents of a CNTLOUT= file**

Here is an edited sample PROC CONTENTS done on a CNTLOUT= file created for a simple numeric format:

The CONTENTS Procedure

<table>
<thead>
<tr>
<th>Data Set Name</th>
<th>WORK.NUM2LETTER_CDS</th>
<th>Observations</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>DATA</td>
<td>Variables</td>
<td>21</td>
</tr>
<tr>
<td>Engine</td>
<td>V9</td>
<td>Indexes</td>
<td>0</td>
</tr>
<tr>
<td>Created</td>
<td>Sunday, July 27, 2008 09:21:35 AM</td>
<td>Observation Length</td>
<td>136</td>
</tr>
<tr>
<td>Last Modified</td>
<td>Sunday, July 27, 2008 09:21:35 AM</td>
<td>Deleted Observations</td>
<td>0</td>
</tr>
<tr>
<td>Protection</td>
<td>Compressed NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Set Type</td>
<td>Sorted NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>DATATYPE</td>
<td>Char</td>
<td>8</td>
<td>Date/time/datetime?</td>
</tr>
<tr>
<td>18</td>
<td>DECSEP</td>
<td>Char</td>
<td>1</td>
<td>Decimal separator</td>
</tr>
<tr>
<td>7</td>
<td>DEFAULT</td>
<td>Num</td>
<td>3</td>
<td>Default length</td>
</tr>
<tr>
<td>19</td>
<td>DIG3SEP</td>
<td>Char</td>
<td>1</td>
<td>Three-digit separator</td>
</tr>
<tr>
<td>16</td>
<td>EEXCL</td>
<td>Char</td>
<td>1</td>
<td>End exclusion</td>
</tr>
<tr>
<td>3</td>
<td>END</td>
<td>Char</td>
<td>16</td>
<td>Ending value for format</td>
</tr>
<tr>
<td>12</td>
<td>FILL</td>
<td>Char</td>
<td>1</td>
<td>Fill character</td>
</tr>
<tr>
<td>1</td>
<td>FMTNAME</td>
<td>Char</td>
<td>32</td>
<td>Format name</td>
</tr>
<tr>
<td>9</td>
<td>FUZZ</td>
<td>Num</td>
<td>8</td>
<td>Fuzz value</td>
</tr>
<tr>
<td>17</td>
<td>HLO</td>
<td>Char</td>
<td>11</td>
<td>Additional information</td>
</tr>
<tr>
<td>4</td>
<td>LABEL</td>
<td>Char</td>
<td>1</td>
<td>Format value label</td>
</tr>
<tr>
<td>21</td>
<td>LANGUAGE</td>
<td>Char</td>
<td>8</td>
<td>Language for date strings</td>
</tr>
<tr>
<td>8</td>
<td>LENGTH</td>
<td>Num</td>
<td>3</td>
<td>Format length</td>
</tr>
<tr>
<td>6</td>
<td>MAX</td>
<td>Num</td>
<td>3</td>
<td>Maximum length</td>
</tr>
<tr>
<td>5</td>
<td>MIN</td>
<td>Num</td>
<td>3</td>
<td>Minimum length</td>
</tr>
<tr>
<td>11</td>
<td>MULT</td>
<td>Num</td>
<td>8</td>
<td>Multiplier</td>
</tr>
</tbody>
</table>
Managing format catalogs

T10. Use PROC CATALOG to manage format “libraries.”

SAS format “libraries” are stored in files that are (more) correctly referred to as catalogs. To manage them, PROC CATALOG is generally preferred over PROC DATASETS. For example, the code below lists all formats in the default work catalog:

```
proc catalog cat=work.formats ;
contents;
run;
```

To delete a format, use code similar to the following (from SASLOG):

```
88 proc catalog cat=work.formats;  * replace ‘work’ with catalog libname;
89 delete num2letter.format;
90 delete num3letter.format;
91 run;
```

NOTE: Deleting entry NUM2LETTER.FORMAT in catalog WORK.FORMATS.
NOTE: Deleting entry NUM3LETTER.FORMAT in catalog WORK.FORMATS.

And the final tip is:

T11. Skillful use of PROC FORMAT, especially the CNTLIN= and CNTLOUT= options, can improve your programs.

References


Contact Information

Thomas E. Billings
Kaiser Permanente
1950 Franklin; 14th floor
Oakland, CA 94612

Phone: 510-987-1320
Email: tebillings@yahoo.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.