

Picture This: Some Useful Formats

Rob Krajcik



HASUG 20-Aug-2009

Copyright © 2007 SAS Institute Inc., Cary, NC, USA. All rights reserved. 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 registered trademarks or trademarks of their respective companies. All Rights Reserved.

Proc Format -- Picture

- **Picture Statement Creates a template for printing numbers**
- **PICTURE *name* <(format-option(s))>**
<value-range-set-1 <(picture-1-option(s))>
<value-range-set-2 <(picture-2-option(s))>
...
<value-range-set-n <(picture-n-option(s))>
- **The *name* must be a SAS name up to eight characters long, not ending in a number.**
- ***format-options*: DEFAULT=, FUZZ=, MAX=, MIN=, NOTSORTED, DATATYPE={DATE | TIME | DATETIME}, DECSEP=, DIG3SEP=, FILL=, MULTILABEL, MULTIPLIER (MULT)=, NOEDIT, PREFIX, ROUND**
- ***value-range-set*: specifies one or more variable values and a template for printing those values.**

Proc Format - *format-options*:

- DATATYPE=DATE | TIME | DATETIME specifies that you can use **directives** in the picture as a template to format date, time, or datetime values.
- FILL='**character**' specifies a character that completes the formatted value.
- MULTIPLIER=**n** specifies a number that the variable's value is to be multiplied by before it is formatted.
- NOEDIT specifies that numbers are message characters rather than digit selectors; that is, the format prints the numbers as they appear in the picture.
- PREFIX='**prefix**' specifies a character prefix to place in front of the value's first significant digit. You must use zero digit selectors or the prefix will not be used.
 - ROUND rounds the value to the nearest integer before formatting.

Proc Format - directives:

Directives are special characters that you can use in the picture to format date, time, or datetime values:

```
%a -- Abbreviated weekday name
%A -- Full weekday name
%b -- Abbreviated month name
%B -- Full month name
%d -- Day of the month as a decimal number (1-31), with no leading zero
%H -- Hour (24-hour clock) as a decimal number (0-23), with no leading zero
%I -- Hour (12-hour clock) as a decimal number (1-12), with no leading zero
%j -- Day of the year as a decimal number (1-366), with no leading zero
%m -- Month as a decimal number (1-12), with no leading zero
%M -- Minute as a decimal number (0-59), with no leading zero
%p -- Locale's equivalent of either AM or PM
%S -- Second as a decimal number (0-59), with no leading zero
%U -- Week number of the year (Sunday as the first day) as a decimal number
    (0,53), with no leading zero
%w -- Weekday as a decimal number (1= Sunday, 7)
%y -- Year without century as a decimal number (0-99), with no leading zero
%Y -- Year with century as a decimal number
%% -- %
```

Any directive that generates numbers can produce a leading zero, if desired, by adding a 0 (zero) before the directive.

Proc Format - value-range-set:

- Single value (12 or 'A')
- a list of values (0 - 100) or (0 -< 100)
- You can use LOW or HIGH as one value in a range, and you can use the range LOW-HIGH to encompass all (non-missing) values.
- You can use the keyword OTHER as a single value.

Some Examples: timestamp

Requirements:

Timestamp will be in the form YYYYMMDDHHNNSS

Where YYYY = 4 digit year,

MM = 2 digit zero-filled month

DD = 2 digit zero-filled day

HH = 2 digit zero filled hour (24-hr clock)

NN = 2 digit zero filled minute

SS = 2 digit zero filled second

Some Examples: timestamp

```
proc format lib=work;  
    picture nihdtm (default=14) other='%Y%0m%0d%0H%0M%0S'  
    (datatype=datetime);  
run;
```

```
data _null_;  
    dtm = datetime();  
    call symput('timestamp',put(dtm,nihdtm.));
```

```
run;
```

```
%put timestamp = &timestamp;
```

```
timestamp = 20090819150311
```

Some Examples: Phone Numbers

```
proc format lib=work;
  picture phone (default=14)
    low - 1000000000          = 'ERROR' (noedit)
    1000000000 <- 9999999999 = '(999) 999-9999' (prefix='(')
    9999999999-high         = 'ERROR' (noedit)
    other                    = 'MISSING' (noedit)
  ;
run;
```

Some Examples: Phone Numbers

```
data _null_;
    infile cards;
    input phone $char14.;
    phnum=input(compress(phone,compress(phone,'0123456789')),best12.);
    put phone= @25 phnum= phone.;
datalines4;
(203) 677-6125
(413) 525 7520
413 7B2 5712
(860).965.5825
.A
iiii
```

Some Examples:

Phone Numbers

phone=(203) 677-6125

phone=(413) 525 7520

phone=413 7B2 5712

phone=(860) .965.5825

phone=.A

phnum=(203) 677-6125

phnum=(413) 525-7520

phnum=ERROR

phnum=(860) 965-5825

phnum=MISSING

Some Examples: Calculating Percentages

Requirements:

The report will display each percent of total in parentheses next to each count, for percentages greater than 0. If the percent of total is 0.0, then no percentage or parentheses will be displayed.

The system will round each percent of total to one decimal place, except where specified otherwise in individual domain requirements. If the percent of total is less than 0.1%, then “<0.1” will be displayed as the percentage.

Some Examples: Calculating Percentages

```
proc format lib=work;
  picture percnnt (default=7 round)
    0 - 0 = ' ' (noedit)
    0.0 <-< 0.1 = '( <0.1)' (noedit)
    0.1 -< 10 = ' 9.9)' (mult=10.0 prefix='( ')
    10 - 99.9 = ' 99.9)' (mult=10.0 prefix='( ')
    99.9 <- 100 = '(100.0)' (noedit)
  other = 'ERROR' (noedit)
;
run;
```

Some Examples: Calculating Percentages

```
/* Testing */  
data _null_;  
  do  
p=-1, 0, 0.005, 0.0999, 0.5, 1.26, 2, 6.25, 93.75, 7.15, 21.24, 99.98, 100, 101.6  
    result=put(p, percnnt.);  
    put @1 p= @10 result=;  
  end;  
run;
```

Some Examples: Calculating Percentages

p=-1	result=ERROR
p=0	result=
p=0.005	result=(<0.1)
p=0.0999	result=(<0.1)
p=0.5	result=(0.5)
p=1.26	result=(1.3)
p=2	result=(2.0)
p=6.25	result=(6.3)
p=93.75	result=(93.8)
p=7.15	result=(7.2)
p=21.24	result=(21.2)
p=99.98	result=(100.0)
p=100	result=(100.0)
p=101.6	result=ERROR

Conclusions:

- Formats, in general, help you separate your code from data, and reduce code maintenance.
- Picture formats can save the day when it comes to providing custom formats.
- They can also save a lot of coding.

About the Speaker

Rob Krajcik

Principal Analyst II

Bristol-Myers Squibb

5 Research PKWY

Wallingford, CT 06492-7660

(203) 677-6125 (phone)

(203) 677-6197 (fax)

robert.krajcik@bms.com