Setting Dates in a Production Job

Deb Cassidy, Cardinal Health, Inc.

ABSTRACT

A set of production jobs produced several reports. The original code required a programmer to edit several jobs each month to specify the correct fiscal years and month for the report. Editing this monthly job required considerable time and was very prone to errors.

There were numerous occurrences where the value of a variable or the variable name had to represent a specific fiscal year or a month. For example, some reports needed values for a variable to be 'FY96' or FY95'. Others needed variables named Jul95 and Jul96. Titles were required that read "FY 96 vs. 95" or "Summary for March 96".

This paper shows how to use date functions and macros to set all these values without editing a single job.

OVERVIEW

Example 1 the values that were being changed each month. Notice the fiscal year values depending upon whether the month was in the first half or the second half of the year since our fiscal year is July-June.

The code is shown at the end of the paper. Please note the line numbers are for reference only and are not part of the code. The code consists of two pieces. First, the setup macro creates all the necessary values to be passed to the jobs. Second, the %INCLUDE statements bring in the individual jobs. Jobs can easily be added or deleted from the monthly run by changing these statements. By storing each job separately, changes can also easily be made to individual jobs.

SETUP MACRO

Line 1 starts the macro named SETUP. Line 2 creates a dataset named SETUP which will end up with one observation. You could also use DATA _NULL_ which won't actually create a dataset. However, if you do so, eliminate the PROC PRINT at the end. The dataset was intentionally created for verification purposes.

These jobs are always run during end-of-month processing for the previous month. In Line 3, the TODAY function is used to determine the current date from the system date. The INTNX function is used to advance a date by a specified number of time periods. In this case, the number is negative so the date reverses 1 month. Unless otherwise specified, the date will be the first day of the time period. For example, if the job is run on April 6, 1997, reversing the date one month will set the date to March 1, 1997. This means the data are then assumed to be for the month of March 1997.

Line 4 puts today's date into a variable called today. This was included only for validation purposes.

Line 5 use the YEAR function to extract the 4-digit year and Line 6 uses the MONTH function to extract the 2-digit month.

Lines 7 and 8 adjust the year to have the correct value for the fiscal year. If the data are for January-June then the fiscal year corresponds to the actual calendar year. If the data are for July-December then the following calendar year is used as the fiscal year. Line 9 creates the previous fiscal year by subtracting 1 from the current fiscal year that was just created.

A 2-digit calendar year is also required in the code. For March 1997 data, the code would require 96/97 and 95/96 to represent the fiscal year and the previous fiscal year, respectively. The MOD function returns the remainder. In this example, 1997 divided by 100 is 19 with a remainder of 97. Thus, yrc is assigned a value of 97 in line 10. Lines 11 and 12 subtract to get the previous years of 96 and 95.

Lines 13-24 assign the name of the month based on the value extracted for month earlier in the code.

Lines 25-33 uses CALL SYMPUT to create macro variables that can then be passed to the

report programs. The name of the macro variable to be created is the first parameter and is in quotes because it is a value and not a variable itself when it is part of the CALL SYMPUT. The second parameter is the variable that holds the value that will become the macro variable.

By default, numeric values are right-justified. Since macro variables are always character, your macro variables created from numeric values will have lead blanks. To eliminate this problem, use the LEFT function.

Since the length of mnthname must be 9 to accommodate the longest month, there will be trailing blanks in the shorted months. The name of the month will be inserted within a title so the trailing blanks must be eliminated. This is done with the TRIM function. The other macro variables will also have trailing blanks but they won't interfere with the rest of the code.

Lines 35 & 36 are included for validation purposes only. If you don't use the FORMAT statement, the dates will be printed as SAS date values.

Line 38 ends the macro. Line 39 actually runs the macro.

Lines 41 is also included for validation purposes only. These options can generate hundreds of pages of code so they should be removed once you are sure all the code is working.

Lines 42-47 brings in each program and runs it.

An & precedes the macro variable name in the code. See the sample lines of code at the end of the paper.

SUMMARY

By using macros and a few date functions, the entire processing for running these jobs now takes less time than the editing process took originally. It also everything is created correctly because the possibility of typographical errors have been eliminated.

REFERENCES

SAS Institute Inc., SAS® Guide to Macro Processing, Version 6, Second Edition, Cary, NC: SAS Institute Inc., 1990. 319 pp.

SAS Institute Inc., SAS® Language: Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc., 1990. 1042 pp.

CONTACT

Deb Cassidy
Senior SAS Programmer/Analyst
Cardinal Health, Inc.
5555 Glendon Court
Dublin, OH 43016

(614) 717-7136 (voice)
(614) 717-8136 (fax)
Dcassidy@cardhealth.com (e-mail)

EXAMPLE 1 - VALUES FOR TEXT CHANGES

	Report for September 96	Report for March 97
Month	September	March
Current Year including century	1996	1997
Previous Year including century	1995	1996
Current Fiscal Year	97	97
Previous Fiscal Year	96	96
Year for Jul-Dec of current fiscal year and Jan-Jun of previous fiscal year	96	96
Year for Jan-Jun of current fiscal year	97	97
Year for Jul-Dec of previous fiscal year	95	95

CODE

```
1     %macro setup;
2     data setup;
3         datamnth=intnx('month',today(),-1);
4         today=today();
5         datayr=year(datamnth);
6         datamon=month(datamnth);
7          if 1<=datamon<=6 then yrclong=datayr;
8     else if 7<=datamon<=12 then yrclong=datayr+1;
9          yrblong=yrclong-1;
10         yrc=mod(yrclong,100);
11         yrb=yrc-1;
12         yra=yrc-2;

13         if datamon=1  then mnthname='January  ';
14    else if datamon=2  then mnthname='February ';
15    else if datamon=3  then mnthname='March    ';
16    else if datamon=4  then mnthname='April    ';
17    else if datamon=5  then mnthname='May      ';
18    else if datamon=6  then mnthname='June     ';
19    else if datamon=7  then mnthname='July     ';
20    else if datamon=8  then mnthname='August   ';
21    else if datamon=9  then mnthname='September';
22    else if datamon=10 then mnthname='October  ';
23    else if datamon=11 then mnthname='November ';
24    else if datamon=12 then mnthname='December ';

25        call symput('yra',left(yra));
26        call symput('yra',left(yra));
27        call symput('yrb',left(yrb));
28        call symput('yrc',left(yrc));
29        call symput('yrblong',left(yrblong));
30        call symput('yrclong',left(yrclong));
31        call symput('runyear',left(datayr));
32        call symput('mnthname',left(trim(mnthname)));
33        call symput('runmonth',left(datamon));
34    run;

35    proc print data=setup;
```

```
36      format today datamnth date7.;
37     run;
38     %mend setup;

39     %setup;

40     ** only use these options when debugging code;
41     options symbolgen mprint mtrace;

42     %include 'l:\sasprdtn\mj103a.sas';
43     %include 'l:\sasprdtn\mj103b.sas';
44     %include 'l:\sasprdtn\mj103c.sas';
45     %include 'l:\sasprdtn\mj103d.sas';
46     %include 'l:\sasprdtn\mj103e.sas';
47     %include 'l:\sasprdtn\mj103f.sas';
```

SAMPLE LINES OF CODE FROM A JOB

```
title3 "Summary for &mnthname &runyear";


array old (*)     jul&yra aug&yra sep&yra oct&yra nov&yra dec&yra
                  jan&yrb feb&yrb mar&yrb apr&yrb may&yrb jun&yrb tot&yrb ytd&yrb;
array new (*)     jul&yrb aug&yrb sep&yrb oct&yrb nov&yrb dec&yrb
                  jan&yrc feb&yrc mar&yrc apr&yrc may&yrc jun&yrc tot&yrc ytd&yrc;
array o (*)       jul     aug     sep     oct     nov     dec
                  jan     feb     mar     apr     may     jun     ytot    ytdt;


nme = "&yrclong";
```