What Fiscal Year is this and when does it start and end?

Wayne Finley, Health and Welfare Data Center, Sacramento, CA

ABSTRACT

Until SAS® version 6.07, I usually wrote several code lines to compute what was a date's fiscal year start and end dates. If a SAS date was in a current Year first six months the State of California's fiscal year started in the prior year's July 1st and ended in the current year June 30th. When the date is the last 6 months, then the fiscal year started in the current year's July 1st and ended the next year on June 30th. The resultant code, besides being hard to read, in several places we found it created Y2K errors. This paper demonstrates the INTNX function's new features to shift the interval starting point from normal start point to another staring point. Specifically the paper shows shifting the year start point from Jan 1st to July 1st (the state's fiscal year start).

INTRODUCTION

The Cost accounting systems, which we maintain at the Heath & Welfare Data Center, depend on computing the fiscal year for our monthly processes. Prior to SAS® version 6.07and 6.11, several lines of code were needed to determine the fiscal year of a SAS date. This paper will show New INTNX features to allow one to compute a fiscal year start and end Dates for a given SAS date. The INTNX function computes the start/end dates for an interval of date/time period. This paper's scope will be on the computing of year interval and how to shift those year intervals from the normal start and end of a year (Jan 1 and Dec 31) to a different start and end such as July 1 and Jun 30th. I only found this feature in SAS Technical Report P-222 for Release 6.07. Also I demonstrate INTNX's new parameter option which computes a time interval's beginning, end, or middle time point.

I have used these new features to correct Y2K code errors and lessen the number lines to be read by future maintenance programmers.

The paper will present real-life examples of SAS code showing the old code and the new and improved code.

THE INTNX FUNCTION DEFINITION

INTNX advances a date, time, or datetime value by given interval.

SYNTAX:

INTNX (*'interval<n><.s>',from, to ,<'alignment'>*)

DESCRIPTION:

For the *interval* parameter, one can now change the interval to a multiple of the base period by using the *n* value. For example, 'YEAR2' indicates a two-year interval. The default value for *n is* 1. The .S specifies that the intervals are shifted to later starting points. The value .S represents a subperiod of the base period. For example: 'Year.3' specifies a yearly period, which starts on March 1 and ends on the last day of February of the next year.

The new *alignment* argument now allows you to advance dates to the midpoint or the end of an interval. Previously, returned date values were only the beginning interval date. The values of *alignment* are Beginning, Middle, and End. The aliases are B, E, and M respectively.

SHIFTED YEAR INTERVALS

To shift the beginning point of the year to a fiscal year, which starts in July, you specify 'YEAR.7'. If you wanted to create a period for the U. S. Election, you specify 'YEAR4.11'. This creates a 4-year period, which starts in November of the 1st year.

When you shift a date interval by a subperiod, the shifting subperiod must be less than or equal to the number of subperiods in the interval.

FISCAL YEAR EXAMPLES

EXAMPLE 1:

Example 1 purpose is to compute a fiscal year Macro variable to use in a SAS dataset name. Fiscal Year format is yyxx where yy is the year of 1st six months and xx is year of last six months. For examples: 9798, 9899, and 9900.

The following is the OLD SAS Code:

```
data _null_ ;
  date = intnx('month',today(),-1);
  billmon = put(date,mmddyy6.);
  m = substr(billmon,1,2);
  y = substr(billmon,5,2);
  if m < '07' then
       date1 = intnx('year',date,-1);
  else
       date1 = intnx('year',date,1);
  date2 = put(date1,mmddyy6.);
  y1 = substr(date2,5,2);
  if m < '07' then
    fy = trim(y1) || trim(y) || 'n';
  else
    fy = trim(y) || trim(y1) || 'n';
  call symput('fyr',put(fy,$5.));
  call symput('thismth',put(date,monyy.));
run;
**********;
data rev&fyr;
 revenue.rev& set fyr;
```

Now here is the NEW SAS Code :

```
data _null_;
date = intnx('month',today(),-1);
y1=put(intnx('year.7',date,0,'b'),year2.);
y2=put(intnx('year.7',date,0,'e'),year2.);
fy = y1 || y2 || 'n';
  call symput('fyr',put(fy,$5.));
  call symput('thismth',put(date,monyy.));
run;
data rev&fyr;
  set revenue.rev&fyr;
```

As you can see there several less lines of code in the new way and I think much easier to read. .

EXAMPLE 2.

This example computes a Fiscal Year macro variable to be used in a Report Title. Fiscal year's format is 'YY/XX'. For example: 97/98 98/99 99/00.

Here is the Old SAS code:

```
data _date_;
      date = intnx('month',today(),-1);
      billmon = put(date,mmddyy6.);
      m = substr(billmon,1,2);
      y = substr(billmon,5,2);
      if m < '07' then
        date1 = intnx('year',today(),-1);
      else
        if m = '12' then
          date1 = intnx('year',today(),0);
      else
          date1 = intnx('year',today(),1);
      billmon1 = put(date1,mmddyy6.);
      y1 = substr(billmon1,5,2);
```

Example 2 - OLD SAS Code continued:

```
if m < '07' then
      do;
        fy = trim(y1) || trim(y);
        fyear = trim(y1) || '/' || trim(y);
      end;
    else
      do;
        fy = trim(y) || trim(y1);
        fyear = trim(y) || '/' || trim(y1);
      end;

  format date  monyy7. ;
  put _all_ ;
     call symput('fiscal',put(fyear,$5.));
  run ;
date=jun1999
y1=98 fy=9899 fyear=98/99
macro variable fiscal resolves to 98/99
```

Example 2 New SAS code:

```
    data _null_;
       date =intnx('month',today(),-1);
       y1=put(intnx('year.7',date,0,'b'),year2.);
*fiscal start;
       y2=put(intnx('year.7',date,0,'e'),year2.); *
fiscal end ;
       m=put(date,mmddyy2.);
       fyear= y1 || '/' || y2;
    format date mmddyy10. ;
    put _all_ ;
    call symput('fiscal',fyear);
     run;
date=06/01/1999 y1=98 y2=99 m=06
fyear=98/99
macro variable fiscal resolves to 98/99
```

Again, a lot less code is needed. Both these examples also used the put function, and the year format YEAR2 to convert fiscal year start and end dates to a 2-character string for the Year. This is okay for y2k, as long the year variables are not used for computing.

EXAMPLE 3. A Y2K ERROR

The following macro example was found in a modification made to HWDC's MVS Cost Recovery system, MICS®.

```
%macro  setyrst(edate) ;
        enddt = &edate ;
        tempmnth = month(enddt);
        if tempmnth > 6 then
          yrstart = mdy(07,01,(year(enddt))-1900);
        else
          yrstart = mdy(07,01,(year(enddt))-1901);
%mend setyrst;
```

The code works fine until year 2000. The problem occurs when 1900 is subtracted from the year value. A 3 digit value Of 100 occurs when the year is 2000. MDY only works on 2 digit or 4digit values. This is the resultant SAS log for three test dates today, 01/26/2000 and 07/26/2000:

1st Test using today (7/26/99):

```
data _null_ ;
    %setyrst(today()) ;
SYMBOLGEN:  Macro variable EDATE resolves to today()
YRSTART=JUL1999
```

2nd Test using 1/26/2000:

```
 data _null_ ;
  %setyrst('26jan2000'd) ;
SYMBOLGEN:  Macro variable EDATE resolves to
'26jan2000'd
YRSTART=JUL1999
```

3rd test using 07/26/2000:

```
 data _null_ ;
 %setyrst('26jul2000'd) ;
SYMBOLGEN:  Macro variable EDATE resolves to
'26jul2000'd
NOTE: Invalid argument to function MDY
YRSTART=.
ENDDT=26JUL00 TEMPMNTH=7 YRSTART=.
```

The third test cause the error to occur. The Year value in the MDY function was a 3digit number (100) .

Example 3 with New SAS code:

```
%macro  setyrst(edate) ;
  data _null_ ;
    yrstart= intnx('year.7',&edate,0) ;
    format yrstart mmddyy10. ;
     put yrstart = ;
 %mend setyrst;
%setyrst(today()) ;
SYMBOLGEN:  Macro variable EDATE resolves to
today()
YRSTART=07/01/1999
%setyrst('26jan2000'd) ;
SYMBOLGEN:  Macro variable EDATE resolves to
'26jan2000'd
YRSTART=07/01/1999
 %setyrst('26jul2000'd) ;
SYMBOLGEN:  Macro variable EDATE resolves to
'26jul2000'd
YRSTART=07/01/2000
```

No errors occur with the new code and again I wrote less code lines.

CONCLUSION

Using the new INTNX Features reduces the number code lines written and makes a more readable program. Less lines of codes makes for better understanding. Future maintenance programmers will thank you if you use these new Features.

Note: SAS is a registered trademark of SAS Institute, Cary, NC, and USA. MICS is a registered trademark of Computer Associates Inc.

REFERENCES

Andrew H. Karp,
"Working with SAS © Date and Time Functions"
Proceedings of the Twenty Fourth Annual SAS users Group International Conference, Cary,NC: SAS Institute., 1999

SAS Institute, Inc., SAS ® *Technical Report P222, Changes and Enhancements to Base SAS® Software Release 6.07,Cary, NC: SAS Institute, INC., 1991*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. I can be contacted at:

Wayne Finley
Health and Welfare Agency Data Center
1651 Alhambra Blvd.
Sacramento, CA 95816-7092
Work Phone: (916) 739-7723
Fax: (916) 739-7773
Email: wfinley@hwdc.state.ca.us