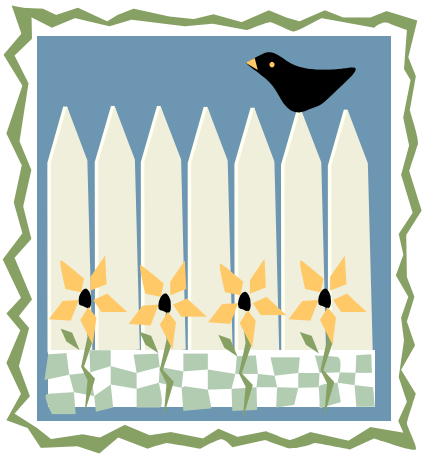
Pretty Dates All in a Row

Dianne Louise Rhodes,
Westat,
Rockville, Maryland



INTRODUCTION

Have you been faced with the task of production reports that have a start and end date that change based on the date the report is run? Here are some tricks and tips for using the system date and the SAS functions INTNX with interval dot notation and INTCK to programmatically produce formats for your dates.

To simplify the running of reports which are scheduled at regular intervals, we can use SAS functions to create macro variables to represent the date ranges for the current execution of the report. This avoids the necessity of changing parameters every time we run. Using these macro variables, I can create formats dynamically which represent the time spans covered in the report.

WORKING WITH SAS DATES

A SAS date, time or datetime variable is a numeric variable which represents the number of days before or after January 1, 1960. The value of a time variable represents the number of seconds after midnight. Valid SAS Dates are from 1582 AD to 20,000 AD. SAS dates can be created using a number of SAS functions. DATE() and TODAY() both return the current system date as a SAS date. The date can also be derived from the SAS macro variable &SYSDATE.

CREATING MACRO VARIABLES

CALL SYMPUT

One common way of creating macro variables is to use CALL SYMPUT in a DATA _NULL_ step to load the value into a macro variable. The syntax is:

```
CALL SYMPUT("VARNAME", CHARACTER STRING VALUE) ;
```

I used CALL SYMPUT to count the number of weeks in the report.

MACRO FUNCTION %SYSFUNC

%SYSFUNC can execute SAS functions or user-written functions in open code. The syntax is

```
%SYSFUNC(function(argument(s), <format>)
```

A common use of %SYSFUNC is to put today's date into a title line:

```
Title "%sysfunc(date(),worddate.) Absence Report";
```

Executing this statement on July 13, 2004 produces this Title Statement:

```
Title "July 13, 2004 Absence Report" ;
```

I used %sysfunc since it did everything that I wanted in one statement. (I admit that I had much help with these; see the Acknowledgements.)

THE PROBLEM

Because I want to customize my date ranges to reflect our Saturday to Friday work week, I needed to learn a little more about shifting intervals using the INTNX function. The solution to this problem evolved from a discussion on SAS-L: "How do I find the last Friday of the month for every month?" It was suggested using the INTNX function with dot notation and the number of the day of the week. For the details of this discussion, see the SAS-L archives.

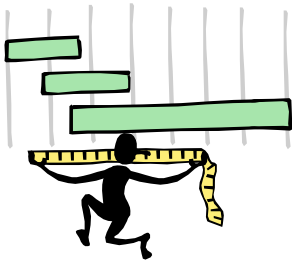
INTNX

The INTNX function advances a date, time, or datetime value by a given interval and returns a date, time, or datetime value. The syntax is:

INTNX(*'interval',start-from,increment*<,'*alignment*'>)

The first parameter is an interval key word enclosed in single quotes. In the examples, you will see that the quotes are not necessary. This can be YEAR, QTR, MONTH, DAY, etc. The second parameter is a SAS date. The third parameter is a number, how many increments to move. The last parameter is an alignment key word, enclosed in single quotes, which can be BEGINNING (or BEG or B), MIDDLE (or MID or M), and END (or E). The default is B (first day of the period.)

INTERVALS



Both the INTNX and INTCK functions are interval functions. Both can use intervals in the form of

Name<multiple><.starting-point>.

Name is the name of the interval (week, year, etc.)

Multiple creates a multiple of the interval. Multiple can be any positive number. The default is 1. For example, YEAR2 indicates a two-year interval.

.starting-point

is the starting point of the interval. By default, the starting point is 1. A value greater than 1 shifts the start to a later point within the interval. The unit for shifting depends on the interval. For example, YEAR.3 specifies a yearly period from the first of March through the end of February of the following year.

For more detail on intervals and shifting, I have included the tables from the on-line documentation as an Appendix.

REPORTING DATE

For my reports, which I usually run on Monday, I want to find last Friday's date. The Week interval starts with Sunday, so Friday is week.6.

```
ReportDate= intnx(week.6, today(),0)
```

The Start Date of the period will be the Saturday of the week before

```
StartDate = intnx(week.7, ReportDate, 0)
```

Since I want to use %sysfunc to put the results into macro variables my actual code looks like:

```
%* The Weekending date is always a Friday (week.6)    ;
%let ReportDate = %sysfunc(intnx(week.6, "&SYSDATE"D, 0),
DATE9.);
%let StartDate = %sysfunc(intnx(week.7, "&ReportDate"D, 0),
DATE9.) :;
```

THE COLUMN FORMAT

The format is created

```
Proc format ;
Value weekcol
Other = "Through Week Prior to &StartDate"
"&StartDate"D - "&ReportDate"D = "Week Ending &ReportDate"
;
```

Note the use of the value "other" in the format. The side effect of this usage is the reporting programs must delete data for dates outside of the range. For example, if I run the report on Wednesday, July 7, I want last week's report, and there was activity for the first part of this week I want to exclude from the report. This is accomplished by checking the date:

```
If CreateDate > "&ReportDate"D then delete ;
```

THE ROW FORMAT

For the cumulative history report, I want a week by week summary. In version 9 there is a function to count weeks, but we are still using 8.2. So I concocted a routine that uses the INTCK function to help me count weeks.

INTCK

The INTCK function counts intervals by using a fixed starting point for the interval as opposed to counting in multiples of the interval unit. Partial intervals are not counted. For example, WEEK intervals are determined by the number of Sundays that begin between the **from** and the **to**, not by how many seven-day periods fall in between the **from** and the **to**. The syntax is:

INTCK('**interval**',**from**,**to***)*

The interval can use the name<multiple><.starting point> form described for INTNX.

Examples

—weekvar=intck('week2.2','01jan97'd, '31mar97'd); put weekvar;6

—wdvar=intck('weekday7w','01jan97'd, '01feb97'd); put wdvar;26

To count the number of weeks to include in the report

```
/* Compute the number of weeks from startdate to reportdate
*/
data _null_ ;
  startdate='18Jul03'd ;
     today=today();
     reportdate=intnx('week.6',today,0) ;
     week=intck('week.6',startdate,reportdate)-1 ;
     call symput('weekfrm',put(week,3.)) ;
 run;
```

The symput statement creates a macro variable &weekfrom, available in the next step.

This proc format generates a format definition with a line for each week from 0 to &weekfrm going backwards from today to the start date.

```
/*  Generates a format for each week from today back to
"&prtref" */
%macro doweek ;

proc format ;
value weekrow
Low -  &ref = "Prior to &prtref."
%do ii = 0 %to &weekfrm ;
  %* Note the minus sign, go backwards to &ref date  ;

  %let RDate = %sysfunc(intnx(week.6,"&SYSDATE"D,-
&ii.),DATE9.) ;
```

```
  %let SDate  = %sysfunc(intnx(week.7,"&RDate"D,0),DATE9.)
;
  "&SDate"D - "&RDate"D  = "&SDate - &RDate"

%end ;
; run;
%mend doweek ;
```

Note the semi colon before the run statement closes the value statement of the PROC FORMAT.

APPLICATION USING TABULATE

In application, this routine is an include file called into a driver program that runs the reports. The reports have a master column that looks like this:

	Prior to 30JAN04		Through Week Prior to 26JUN2004		Week Ending 02JUL2004		Total	
	Time in Minutes		Time in Minutes		Time in Minutes		Time in Minutes	
		ColPctSum		ColPctSum		ColPctSum		ColPctSum
Reviewer								
Axxxx, Bennett	10782	3.2	19726	5.7	115	2.0	30623	4.5
Axxxxxx, Claire	22177	6.7	25618	7.4	105	1.8	47900	7.0
Axxxxxx, Carolyn	3379	1.0	-	-	-	-	3379	0.5
Bxxxxxx, Steve	5	0.0	-	-	-	-	5	0.0

And a row format that looks like this:

	Count of Updates Completed	Count of Reviewers Making Updates	Count of BASE IDs with updates
Time Period			
Prior to 30JAN04	12003	51	5103
31JAN2004 - 06FEB2004	146	5	143
07FEB2004 - 13FEB2004	14	1	14
14FEB2004 - 20FEB2004	48	9	45
21FEB2004 - 27FEB2004	79	8	76
28FEB2004 - 05MAR2004	154	14	150
06MAR2004 - 12MAR2004	142	13	139
13MAR2004 - 19MAR2004	92	11	85
20MAR2004 - 26MAR2004	63	8	61
27MAR2004 - 02APR2004	66	8	62
03APR2004 - 09APR2004	48	5	48
10APR2004 - 16APR2004	161	13	147
17APR2004 - 23APR2004	88	14	83
24APR2004 - 30APR2004	119	12	118
01MAY2004 - 07MAY2004	126	9	123
08MAY2004 - 14MAY2004	164	15	156
15MAY2004 - 21MAY2004	110	12	103
22MAY2004 - 28MAY2004	54	8	54
29MAY2004 - 04JUN2004	147	12	132
05JUN2004 - 11JUN2004	122	19	99
12JUN2004 - 18JUN2004	80	18	70
19JUN2004 - 25JUN2004	13	9	12
26JUN2004 - 02JUL2004	7	4	7
Total	14046	278	7030

CONCLUSION

SAS datetime functions make it easy to create macro variables to represent date ranges for periodic reports.

REFERENCES

Cody, Ron. (2004). *SAS® Functions by Example*. Cary, NC: SAS Institute Inc.

Karp, Andrew H. (1999), "Working with SAS Date and Time Functions, "Proceedings of the 24th SAS Users Group International.

Viergever, William W. (2003), "Tips from the Hood: Challenging Problems and Tips from SAS-L", Proceedings of the 28th SAS Users Group International

SAS-L Archives at http://www.listserv.uga.edu/archives/sas-l.html

SAS Version 8 On-Line Documentation.

Werner, Nina L. (2003) "Date Parameters for Interval Reporting," Proceedings of the 28th SAS Users Group International.

ACKNOWLEDGEMENTS

The author is indebted to my colleagues who participate in SAS-L for their contributions. I am especially indebted to Paul Dorfman for his help with this problem. He provided insight into the use of formats that I would not have realized on my own.

DISCLAIMER: The contents of this paper are the work of the author(s) and do not necessarily represent the opinions, recommendations, or practices of Westat.

CONTACT INFORMATION

Comments, questions, and additions are welcomed.
Contact the author at:
Dianne Louise Rhodes
Westat
1650 Research Blvd.
Rockville, MD 20850
Phone: (301) 315-5977
Email: diannerhodes@westat.com

TRADEMARKS

Appendix
Working with SAS Dates, Times, and Intervals

Interval Tasks				
Return the number of specified time intervals that lie between the two date or datetime values	Interval functions	INTCK	week 2 01aug60 01jan01	1055
Advances a date, time, or datetime value by a given interval, and returns a date, time, or datetime value	Interval functions	INTNX	day 14086 01jan60	

Date and Time Intervals

Definitions

duration
is an integer representing the difference, in number of days, between any two dates or times or datetimes.

interval
is a unit of measurement that SAS can count within an elapsed period of time, such as DAYS, TENDAYS and SEMIMONTHS.

Syntax

SAS provides date, time, and datetime intervals for counting different periods of elapsed time. You can create multiples of the intervals and shift their starting point. Use them with the INTCK and INTNX functions and with procedures that support numbered lists (such as the PLOT procedure). The form of an interval is

name<multiple><.starting-point>

The terms in an interval have the following definitions:

name
is the name of the interval. See the following table for a list of intervals and their definitions.

multiple
creates a multiple of the interval. **Multiple** can be any positive number. The default is 1. For example, YEAR2 indicates a two-year interval.

.starting-point
is the starting point of the interval. By default, the starting point is 1. A value greater than 1 shifts the start to a later point within the interval. The unit for shifting depends on the interval, as shown in the following table. For example,

YEAR.3 specifies a yearly period from the first of March through the end of February of the following year.

Intervals Used with Date and Time Functions						
Category	**Interval**	**Definition**	**Default Starting Point**	**Shift Period**	**Example**	**Description**
Date	DAY	Daily intervals	Each day	Days	DAY3	Three-day intervals starting on Sunday
	WEEK	Weekly intervals	Each Sunday	Days (1=Sunday ... 7=Saturday	WEEK.7	Weekly with Saturday as the first day of the week
	WEEKDAY <**days**W>	Daily intervals with weekend days treated as part of the preceding weekday. **Days** identifies the weekend days by number (1=Sunday ... 7=Saturday). By default, **days**=17.	Each day	Days	WEEKDAY1W WEEKDAY35W	Six-day week with Sunday as a weekend day Five-day week with Tuesday and Thursday as weekend days (W indicates that day 3 and day 5 are weekend days)
	TENDAY	Ten-day intervals (a U.S. automobile industry	First, eleventh, and twenty-first of	TENDAY periods	TENDAY4.2	Four ten-day periods starting at the second TENDAY period

Intervals Used with Date and Time Functions						
Category	**Interval**	**Definition**	**Default Starting Point**	**Shift Period**	**Example**	**Description**
		convention)	each month			
	SEMIMONTH	Half-month intervals	First and sixteenth of each month	SEMIMONTH periods	SEMIMONTH2.2	Intervals from the sixteenth of one month through the fifteenth of the next month
	MONTH	Monthly intervals	First of each month	Months	MONTH2.2	February-March, April-May, June-July, August-September, October-November, and December-January of the following year
	QTR	Quarterly (three-month) intervals	January 1 April 1 July 1 October 1	Months	QTR3.2	three-month intervals starting on April 1, July 1, October 1, and January 1
			April 1			
			July 1			
			October 1			
	SEMIYEAR	Semiannual (six-month) intervals	January 1	Months	SEMIYEAR.3	Six-month intervals, March-August and September-February
			July 1			
	YEAR	Yearly intervals	January 1	Months		

Intervals Used with Date and Time Functions						
Category	**Interval**	**Definition**	**Default Starting Point**	**Shift Period**	**Example**	**Description**
Datetime	Add DT	To any date interval			DTMONTH	
					DTWEEKDAY	
Time	SECOND	Second intervals	Each second	Seconds		
	MINUTE	Minute intervals	Each minute	Minutes		
	HOUR	Hourly intervals	Each hour	Hours		

Single-Unit Intervals

Single-unit intervals begin at the following points on the calendar:

Single-Unit Intervals	
These single-unit intervals	**Begin at this point on the calendar**
DAY and WEEKDAY	each day
WEEK	each Sunday
TENDAY	the first, eleventh, and twenty-first of each month
SEMIMONTH	the first and sixteenth of each month
MONTH	the first of each month
QTR	the first of January, April, July and October
SEMIYEAR	the first of January and July
YEAR	the first of January

Shifted Intervals

Shifting the beginning point of an interval is useful when you want to make the interval represent a period in your data. For example, if your company's fiscal year begins on July 1, you can create a year beginning in July by specifying the YEAR.7 interval. Similarly, you can create a period matching U.S. presidential elections by specifying the YEAR4.11 interval.