

How to Have Your Way with a Date Using SAS®

by
Allan M. Kominek

ABSTRACT

This paper will delve into the mysteries of SAS date values. The author has been supporting users of the SAS system for approximately ten years. During those ten years, it has been his experience that SAS date values are most often misunderstood and underutilized. The paper will illustrate the versatility of SAS date values by sighting examples from working systems that revolve around SAS dates. These examples will include a production application that works in tandem with a job scheduling utility and another application that uses dates to load an array to produce a "spreadsheet-like" report. The paper will also discuss converting dates from external sources and using functions to extract even more information from these date values.

INTRODUCTION

Throughout history keeping track of time in terms of dates has been a common practice for all of the great civilizations. During the Golden Age of Greece, the Athenians referred to a particular year as the year in which so-and-so was archon. Being the archon was a political position and the everyday Athenian usually referred to the archon as that old so-and-so. The Romans measured the passage of time from the mythical founding of Rome by Romulus and Remus. Along came Christianity and the birth of Christ was picked as the point from which to measure the passage of time. It was later found out that, due to errors in the original calendar, Christ was actually born 6 years before Christ. Go figure. My personal opinion is that the monks who were in charge of the original calendar may have been sampling the fruits of their labors and missed a few years. Today we have computers to keep track of the time and the date. Unfortunately, they each have their own method, sic, algorithm, for doing so. For example, what's this deal with Julian dates? It's 94132. Beam me up Scotty. We might as well go to stardates. Then there's SAS. As you are well aware, SAS keeps track of the number of days since January 1, 1960. So September 1, 1994 is 12662. Beam me up Scotty. I don't know about you, but if the teacher asked my child what the date was and she answered 12662, I'm sure mom and dad would be scheduling an appointment with the school psychologist. Actually, 12662 could be interpreted as December 6, 1992, which isn't that far off. Remember those monks? Even so, the date with the psychologist would be on the calendar.

THE MYSTERY REVEALED

How do you make sense out of all of this date stuff? Especially, how SAS is doing the book keeping. After all we're all SAS programmers and this is a SAS conference. The only point of reference that I have is my own experiences with SAS. SAS dates were initially a mind blower. They really took some getting used to. The only thing that helped me keep minimally in touch with reality, I won't say sanity, because that was suspect long before SAS came along, was stepping back and using a broader perspective. To paraphrase Albert Einstein, it's all relative. Measuring the passage of time is all relative to the person doing the measuring. Don't be alarmed. We're not going to get into time dilation at speeds approaching those of light. The SAS System rather than picking the founding of a civilization or of a religion, picked January 1, 1960. What momentous date this represents has to this day remained unclear and may remain shrouded in mystery. In my own mind, that was about the time, my family got the new living room furniture. You know the kind where your mom never let you sit on it until it was covered with that clear plastic and after it was covered, you never wanted to sit on it because any exposed part of your flesh would immediately stick. So in my mind, January 1, 1960 represents the start of the plastic seat cover era or PSC. Of course anything prior to this momentous date is BPSC or Before Plastic Seat Covers and anything after is APSC, Ante Plastic Seat Covers. Ante from the Latin meaning after, not anti which would imply some sort of device for combatting the effects of the seat covers. Believe me, the family would have paid handsomely for such a device, but like antigravity, it was to remain beyond our grasp.

TRANSLATION PLEASE

All dates prior to January 1, 1960 are carried as negative numbers, all dates after are carried as positive numbers. This method of storing dates easily lends itself to mathematical calculations. One of the easiest questions to answer using SAS dates is how many days is it between one date and another. Here's an obvious example, how many days is it between January 1, 1960 and September 1, 1994? The answer is 12,662! That's pretty impressive. What's that? Sounds like Dustin Hoffman in Rainman. Well, yeah, I mean most nonSAS people don't go around worrying about how many days have passed between two dates. Notice I said nonSAS people, not normal people. There's a fine line that we're trying to walk here. Most people are not interested in the fact that 60 days from September 1, 1994 is 12,722 days from January 1, 1960. They would be more impressed if Dustin spouted off October 31, 1994. Translation please! SAS provides the means of translating these cryptic numbers into everyday measures of time through a variety of FORMATS, INFORMATS, and FUNCTIONS. Let's take 12722 and say that we want to display it on a report so that even those uninitiated into the mysteries of SAS can interpret it. By assigning it a FORMAT, SAS will automatically

convert it into one of the more mundane date representations:

Date	FORMAT
10/31/94	MMDDYY8.
10OCT94	DATE7.
October 31, 1994	WORDDATE18.
Monday, October 31, 1994	WEEKDATE29.

SAS even has FORMATS for determining the quarter of the year. There's even one that will express it in Roman Numerals. No, it won't tell you the number of days since the founding of Rome! Although traditionally, Rome was founded in 753 BC, that would be 2713 BPSC, which would be somewhere in the vicinity of -1,000,000 to SAS. Unfortunately, in Release 6.07, SAS can only go back to the 17th century, so the date is a gross estimation.

SAS also has the capability of converting conventional dates to SAS dates via INFORMATS. If an external file has a date stored in it, SAS can read it and convert it into a SAS date value.

Date	INFORMAT
103194	MMDDYY6.
94144	JULIAN5.

SAS also has a number of functions that are not only useful for translating dates, but also for manipulating them. One of the most often problems encountered from a support standpoint is that a date is store as a packed decimal and it is necessary to perform calculations on it or to express it in a different format. The key to solving this problem is to get the date stored as a SAS date value. Once it is in this form, it can be manipulated and formatted at will.

Let's assume that once the packed field is read in, it converts to the numeric 103194 and that we've been creative enough to store it in the numeric variable NUM. It can be converted to a SAS date value by using the following code:

```
DATE=INPUT(PUT(NUM,6.),MMDDYY6.);
```

The variable DATE now contains 12722, the SAS date value for 10/31/94.

SO NOW WHAT?

Well, now that we've gone through the gymnastics of making sure that all of the dates that we're going to need are stored as SAS date values, now what do we do? What kind of neat things can you do with these dates? Here are two true to life examples:

1. A job scheduler within a job scheduler.

Our company has a job scheduler. This piece of software allows people to place jobs into production and to specify dates, times, days of the week, days of the month, etc. for these jobs to be run. Inevitably, SAS jobs are developed that are run on a periodic basis. The most logical thing to do would be to place these jobs into production and schedule them to run during off hours when there is a decreased load on the CPU. Reports produced by this job stream could then be ready when people arrive at work. In fact, there is an existing job stream that runs Monday through Saturday and kicks off a number of reports. The job is extremely efficient. The reports are all based on data read from an extremely large data base. The data base is read once, at about 5:00 A.M., and all the reports are generated and distributed before people arrive at work. In addition to these reports, it was also decided that a series of graphs would also be helpful. The graphs would be needed once a month. The first suggestion was to take the data from one of these reports that are run on the first working day of the month and type it into a PC product to produce the graphs. The other suggestion was to produce the graph using SAS and schedule the job to run once a month. Needless to say, the later suggestion made more sense from an efficiency standpoint, but we were able to do it one better. A separate job would require that the data base be read again. Why read the data base again, when there is an existing job stream that has already read the data? Well, one reason might be that the existing job stream runs every day and the graphs are only to be produced once a month on the first working day of the month. That's when we wrote the job scheduler within the job scheduler.

There are only two facts that have to be taken into account to solve this problem:

1. The job stream does not run on Sunday.
2. The graphs must print on the first working day of the month and Sunday is not a work day.

The first working day of the month could very well be the first of the month, unless it fell on a Sunday, then the first working day of the month would be the second. Simple. Now to take advantage of SAS's ability to manipulate dates.

To determine if the current day is the first of the month, it would help to know today's date. There is an automatic macro variable, &SYSDATE, which contains today's date as a SAS date value. There are also two functions, TODAY() and DATE(), which can be used to place today's date into a SAS variable. For this example, there is no need to create another variable, so using the macro variable will be fine. The next task is to determine if it is the first or second day of the month. The DAY function returns the day of the month from a SAS date value. We will also need to know if the second of the month falls on a Monday. This implies that the first fell on a Sunday and that Monday is the first working day of the month. The WEEKDAY function returns the day of the week from a SAS date value. This function returns a 1 for Sunday, a 2 for Monday, and so on. Before showing you the code for the job scheduler within a job scheduler, there is some SAS coding that does not relate to dates that is essential to making this work. This pseudo job scheduler will not work without the ability to conditionally execute DATA and PROC steps. We do not want to execute the DATA and PROC steps that produce the graphs unless it is the first working day of the month. We've figured out how to determine if it is the first working day of the month, but how do we use this to execute the code for the graphs? A CALL EXECUTE will resolve an argument and issue the resolved value for execution. The argument can be a macro expression or one or more SAS statements or pieces of SAS statements. In this example, all the code to produce the graphs was placed in a macro called MONTHLY and another job step was added to the nightly job stream. That job step contained the following DATA step:

```
DATA _NULL_;

    IF DAY(&SYSDATE)=1 OR
       (DAY(&SYSDATE)=2 AND WEEKDAY(&SYSDATE)=2) THEN
        CALL EXECUTE('%MONTHLY');
```

The macro MONTHLY must be defined prior to the DATA step.

2. Forecasting repetitive activities.

Any manufacturing firm has to be concerned with maintaining its capital equipment. Maintenance is scheduled on a periodic basis. Different pieces of equipment have to be maintained on different schedules. Some equipment must be maintained monthly, some quarterly, etc. Some equipment requires different types of maintenance, i.e., electrical, mechanical, etc. Each task that has to be performed has an associated cost. The cost of maintenance can vary greatly from month to month depending on the number and type of tasks scheduled for that month. The problem that we were faced with was creating a forecast of maintenance dollars by month by type for the next five years.

In trying to understand how the solution was arrived at, an explanation of the files involved is necessary. One file, contains the maintenance activity and the associated cost. The activity is identified by a code number. Another file contains the activity code, the date on which the activity is due to be performed next, the type of maintenance, and two other codes. One of these codes identifies how often the activity has to be performed. There are five categories for these tasks, those that are performed monthly, quarterly, semi-annually, yearly, and every 18 months. The other code is, in a sense, a multiplier. A task that is performed every other month would have an M for monthly and a multiplier of 2 to indicate that the time period between activities is two months.

We decided that the best way to present the data was in a spreadsheet-like report, where the rows represent the different types of maintenance and month would be the column. The "cells" would contain the accumulated cost for that type of maintenance for that month. The trick was getting the data into the appropriate "cell".

We've already been given the date on which the task is next scheduled to be performed and how often the task has to be performed, by using SAS's ability to manipulate dates, getting the data in the correct "cell" should be no problem.

The other feature of SAS that made this report a snap was ARRAYS. Once the cost data was accumulated into an ARRAY, producing the spreadsheet-like report was easy. To illustrate the point, let's look at a section of code that will load the array. By this point in the program, the two files have been read in and the appropriate sorts and merges have taken place. The resulting SAS data set contains the following variables.

CODE - M for monthly
Q for quarterly
SA for semiannual
A for annual
R for every 18 months.
COST - the cost associated with a task.
DUE - the date on which the activity is to be performed next.
FREQ - the multiplier.
MAINT - the type of maintenance to be performed.
E for Electrical
M for Mechanical
O for Other

```

DATA COST; SET COST;
  ARRAY A(3,60) A1-A180;
  * There are 60 months in 5 yrs and 3 types of maintenance;
  IF _N_=1 THEN DO;
    DO K=1 TO 3;
      DO L=1 TO 60;
        A(K,L)=0;
      END;
    END;
  END;
  IF MAINT='E' THEN I=1;      * Each type of maintenance is;
  ELSE IF MAINT='M' THEN I=2; * assigned a row in the array;
  ELSE IF MAINT='O' THEN I=3;
  * Measure from the first of the current month;
  START=MDY(MONTH(TODAY()),1,YEAR(TODAY()));
  * determine which column of the spreadsheet to place the cost
  in based on when the task is due and the first of this
  month;
  MONTHDUE=MONTH(DUE);
  J=((MONTHDUE-MONTH(START))+1)+(12*YEAR(DUE)-YEAR(START));
  IF J>0 THEN A(I,J)=A(I,J)+COST; * Accumulate cost;
  * Determine next column based on CODE and multiplier;
  * iterate for 60 month time frame;
  DO WHILE(J < 61);
    IF CODE = 'M' THEN NEXT = FREQ;
    ELSE IF CODE = 'Q' THEN NEXT = FREQ * 3;
    ELSE IF CODE = 'SA' THEN NEXT = FREQ * 6;
    ELSE IF CODE = 'A' THEN NEXT = FREQ * 12;
    ELSE IF CODE = 'R' THEN NEXT = FREQ * 18;
    J=((MONTHDUE-MONTH(START))+1)+(12*((YEAR(DUE)-
    YEAR(START))+NEXT));
    IF J < 61 AND J > 0 THEN A(I,J) = A(I,J) + COST;
    MONTHDUE = MONTHDUE + NEXT; * New month due;
    OUTPUT;
  END;
  RETAIN A1-A180; * For accumulating cost;

```

CONCLUSION:

SAS stores dates as the number of days form January 1, 1960. Mathematical calculations can easily be performed on SAS date values. The SAS System provides the ability to convert dates from various formats into SAS date values. There are also a number of functions for extracting the month, day of the week, and other date related information from SAS date values. You can have your way with a SAS date value.

SAS and SAS/AF are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.