Paper p58-25

A Beginners Guide to SAS® Date and Time Handling

Wayne Finley, State of California HHSDC, Sacramento, CA

Abstract

The SAS system provides a plethora of methods to handle date and time values. Correct date computation became a very important subject when the world became aware of the Year 2000 issue. Computer users now realize most applications contain date and time variables. This beginning tutorial describes how the SAS system 1) stores dates, datetimes, and times; 2) reads date/time variables from "Raw Data Files" and from SAS data sets. It covers when and where to use SAS Informats and formats. Next it describes the easy methods to perform date/time arithmetic via date/time SAS functions.
The paper shows how SAS date Formats can be used in SAS Procedures such as PRINT, FREQ, and CHART.
Finally the tutorial will discuss Year 2000 issues and how SAS software helps you maintain correct date integrity and prevent future Y2k problems in the new millennium.

Introduction

During the New Year's first week I told a colleague what my presentation's title was. And he replied, "Gee isn't that 6 months too late? " Taking umbrage I said the topic of date/time handling was broader and more important than just the infamous "Y2K" issue. My goal will be to convince you this is true. I also mentioned the paper to a computer salesman I know. And he did not understand how one could possibly talk for an hour on just SAS dates and time computation. Well I assured him I could talk for hours on this topic.
This tutorial's purpose is to demonstrate the powerful SAS tools for Date/Time computations and to teach you how to leverage and exploit these tools. Using this knowledge your SAS code will become easier to write, easier to understand, and easier to maintain.

Definition of SAS System Dates and Times

Our current Gregorian calendar is based on history and artificial convention. The year length of almost 365.25 days is accomplished by having normal year be 365 and every 4th year (leap year) be 366. And of course years divisible by 100 are not leap years unless they are divisible by 400. This means 2000 is a leap year and 1900 was not. With these calendar conventions date computation is nontrivial. I liken it to doing non-metric English pound and shilling arithmetic and having to convert to dollars every so often. It's not easy to compute "how many weeks until Christmas" or "which day of the week is payday?" Also, if you worked for the parole board you might want to determine for a set of life prisoners: "how many months until their next parole hearing?". In order to manage these computations SAS adopted a strategy of regarding dates and times as simple numbers.
The SAS system stores dates as the number of elapsed days since January 1,1960. For example, January 1, 1960 is stored as 0.
December 30, 1959's SAS date equals -2, and December 31, 1960 SAS date is stored as 365.

Times are stored as the number of elapsed seconds since midnight.

SAS also allows you to work with composite datetimes. These are stored as elapse seconds since midnight of January 1, 1960.

SAS dates are valid from January 1, 1582 to December 31, 20,000.

Note: Dates before January 1, 1960 are negative integers and after January 1,1960 are positive integers. SAS dates are stored as simple integer numbers. SAS time/datetimes are stored as decimal numbers to include 10th and 100th of seconds. For time values the integer part is the number of seconds from midnight. For datetime the integer part equals the number of seconds from January 1,1960.

The SAS system stores dates, times and datetimes as integer numbers against a common reference point. But one cannot understand dates/times as 13055 or –1042. We need to see dates and times represented by our conventions. To handle these convention transforms, the SAS system provides informats for input, formats for output, and literal date/time constants in the SAS code.

Creating SAS Date Variables

You can create SAS date/time variables three common ways. Using informats, you can translate text data inputs to SAS date/time variables. With SAS date/time constants you create SAS date/time variables directly. And finally you can translate SAS variables, either numeric or character, by invoking SAS functions.

Reading Dates/TImes from External Files

Using SAS informats you can easily input dates/times from external files. I counted at least 24 different informats for date, time, and datetime conventions in the SAS on-line help. Some of the most common ones are as follows:
DATEw. , DATETIMEw ., DDMMYYw., JULIANw., MMDDYYw., TIMEw. and , YYMMDDw.
w represents the format width.
Here is an example:

```
data test ;
 input @1  date date9.  @12 time time5.  @18 datetime
datetime18. ;
cards;
04jul1776  12:00 04jul1776:12:00
11dec1941  23:00 11dec1941:23:00
20apr1971  00:00 20apr1971:00:00
25dec2000  15:01 25dec2000:15:01
 run;
proc print data=test;
run;
```

This gives the following output:

OBS	DATE	TIME	DATETIME
1	-67019	43200	-579039840
2	-6595	82800	-569725200
3	4127	0	356572800
4	14969	54060	293375660

Coding SAS Date/Time Literal Constants

SAS provides a convenient way to declare a date/time constants in programs. You can code a SAS date constant or a SAS time constant by writing date or time enclosed in single or double quotes followed by D (date) , DT (datetime) , or T(time) to indicate value type. Use the following patterns:

```
'ddmmm<yy>yy'D
'hh:mm<:ss.s>T
'ddmmm<yy>yy:hh:mm<:ss.ss>'dt
```

Here are some examples:

```
fdate= '25jan1999'D ;
      if fdate < '01jul99'D ;
      btime = '01:05'T ;
      if datestamp gt '01jan2000:00:00:01'dt;
```

Using date literals with macros variables gives you a powerful way to write easy to maintain SAS programs.
A Macro example;

```
      %let bill_month = '01dec1999'd ;
      run ;
      data invoices:
        set acc.newbills ;
        if install_date < &bill_month ;
```

Transforming SAS Variables to SAS Date/Time Variables

SAS numeric or text variables can be converted to SAS dates by using the INPUT, and PUT functions or by invoking the MDY function.

Using the INPUT Function

Use the INPUT when you have a SAS text variable in a DATE informat convention. You can transform it to a SAS date by invoking the INPUT function with the appropriate informat:

```
    data mdytest;
     begin_sugi = input ('4/9/2000',mmddyy8.) ;
     tutor_date = input ('20000410',  yymmdd8.) ;
     end_sugi  = input ('12apr2000', date9.) ;
     format begin_sugi end_sugi tutor_date  mmddyy10. ;
      put _all_ ;
run;

begin_ sugi=04/09/2000
tutor_date=04/10/2000
end_sugi=04/12/2000
```

Using the PUT and the INPUT Functions Together

The INPUT function is used to convert characters to numeric. The PUT function is used to convert numerics to characters. To convert a numeric variable to a SAS date variable first use the PUT function to transform it to text and then call the INPUT function to complete the conversion to the SAS date value. Remember the SAS dates are special kinds of numeric data.

Here is a numeric to SAS date example:

```
    data _null_  ;
      x=123199 ;
      xchar = put(x,z6.) ;
      xdate = input (xchar, mmddyy6.) ;
      format xdate date7. ;
      put  _all_ ;
   run;

x=123199 xchar=123199 xdate=31DEC99
```

Using the MDY Function

Many times a SAS data set contains separate variables for the day, month, and the year of a date. When this occurs, use the MDY function to create a SAS date. The MDY form is:

```
new_date = mdy( month_var ,day_var,year_var);
```

where :
new_date = name of new sas date variable
month_var =number between 1 and 12
day_var = number between 1-31
year_var= a 1,2 digit number between 00 and 99
or a 4,5 digit number between 1589 and 20000

Examples using the MDY function:

```
data ;
      taxday= mdy(4,15,2000) ;
      fileday = taxday      ;
     put taxday= taxday   worddate.   ;
run ;

      taxday=14715     April 15, 2000
```

This year's taxday SAS date equals 14715.
Here is another MDY example:

```
data mdytest;
    taxmonth= 4 ;
    taxday = 15 ;
    taxyear= 00 ;
taxday=  mdy(taxmonth,taxday,taxyear) ;
put taxday=   taxday yymmdd10.;
run;
```

taxday=14715 2000-04-15
Notice that you can code either a constant or a variable in the parameters passed to the MDY function.
If all observations of a data set happened in 1999 , you could write :

```
occured_date = MDY(o_month,o_day,1999) ;
```

Format Representation of SAS DATE/TIME Variables

As we saw in the the INFORMAT example , we need externals format to output /display SAS date/time variables.
There exists over 30 SAS Formats for date, time, and datetime variables. Both Informats and Formats are documented in the SAS Language Reference manual and in the On-line SAS help files for version 7 and 8.
To display the SAS date value = 14798, you could invoke these commonly used formats:

Format:	Result:
MMDDYY8.	07/04/00
DDMMYY10.	04/07/2000
DATE9.	04JUL2000
MONYY5.	JUL2000

A PROC PRINT Example:

```
proc print data=test noobs ;
format date worddate18.
       time hhmm10.2
     datetime  datetime40.2 ;
run ;
```

Results in this output:

DATE	TIME	DATETIME
July 4, 1776	12:00.00	04JUL1776:12:00:00.0
December 11, 1941	23:00.00	1DEC1941:23:00:00.00
April 20, 1971	0:00.00	20APR1971:00:00:00.00
December 25, 2000	15:01.00	25DEC2000:15:01:00.00

Arithmetic Date/Time Calculations

You can easily compute the number of days between 2 dates by subtracting one SAS date from another. This result can then be divided by the time period. For example to determine the number of years between 2 dates code the following:
Years = (date1-date2) / 365.25 ;
you can use 30.4 as the time period to calculate number of months between sas dates.
The following is SAS code for computing ages in years:

```
options yearcutoff=1940;
   data testdate;
     input @01 name $10.
           @12 birthdate mmddyy6.
           @20 startdate yymmdd10.;
     startage=int((startdate-birthdate)/365.25);
     agenow  =int((date()-birthdate)/365.25);
     format birthdate startdate monyy7. ;
datalines;
gretchen   120574  1997/12/23
mike       092547  1966/06/15
wayne      092040  1966/08/01
run;
  proc print data=testdate;
run;
```

The results are:

name	birthdate	startdate	startage	agenow
Gretchen	dec1974	dec1997	23	25
Mike	sep1947	jun1966	18	52
Wayne	sep1940	aug1966	25	59

SAS Date/Time Functions

There over 20 useful functions to handle dates. You can get the current date or datetime with these functions. You may extract "parts" from a SAS date, time, or datetime. And you can finally perform special date arithmetic with 2 functions.

Obtaining the Current System Date/Time

You create SAS variables for the current date and time as shown by the following SAS code:

```
data today;
  a= date( ) ;
  b= today( ) ;
  c= time( ) ;  d= datetime( ) ;
```

```
    put a= date. b= mmddyy8. c= time10.2   d= datetime16. ;
    run;

a=09JAN00  b=01/09/00 c=22:46:57.2  d=09JAN00:22:46:57
```

Extracting 'Parts' From a SAS Date Variable

To obtain parts from a SAS date use the following functions:

MONTH Returns the month number
DAY Returns the day of the month
YEAR Returns the year
QTR Returns the quarter of the year
WEEKDAY Returns the day of the week (Sunday=1)

Example :
Assume a SAS data set contains a variable called taxdate. Invoking the date functions results in following for April 15,2000. For example:

```
data extract;
        taxday = '15apr2000'd ;
        tmonth= month(taxday) ;
        tday = day(taxday) ;
        tyear= year(taxday) ;
        tqtr= qtr(taxday) ;
        twday= weekday(taxday) ;
        format taxday mmddyy10. ;
        put _all_ ;
      run ;

taxday=04/15/2000 tmonth=4 tday=15 tyear=2000 tqtr=2
twday=7
```

Extracting 'Parts' From a SAS Time Variable

You can also get parts from a SAS time value by using the HOUR, MINUTE, and SECOND functions. These are obtained in the same manner as the date functions discussed above.

Extracting the DATE/TIME Parts of a Datetime Variable

Using the DATEPART and TIMEPART functions you can easily get the date or the time value of a Datetime variable. The following example demonstrate these functions:

```
data extract;
       now=datetime() ;
       now_date= datepart(now) ;
       now_time= timepart(now) ;

    format  now datetime18.
              now_date mmddyy10.
             now_time time10.2
                        ;
          put _all_ ;
     run ;
```

Results of extracts are:

```
 now=09JAN00:23:36:47 now_date=01/09/2000
now_time=23:36:47.0
```

Converting to Julian Date

If you need or want to use Julian dates, the JULDATE (sasdate) function returns the Julian date from a given SAS date; the DATEJUL (Juliandate) function returns a SAS date from a given Julian date. For example:

```
   data julian ;
        juldate=juldate('09apr2000'd);
        datejul=datejul(juldate) ;
       put juldate=  z5. datejul= yymmdd10. ;
       run ;

   juldate=00100 datejul=2000-04-09
```

Computing Intervals between SAS Dates/Times

Difference functions are extremely powerful and useful. There are two of them:

INTCK(time_interval,from, to)

Computes the number of time intervals(i.e. 'days', ;weeks', 'months', 'years') between the "from" and the "to" date, time, datetime values.

The INTCK function counts the intervals from a fixed time beginning point, not in multiples intervals from the 'from' value. I.E. No partial intervals are counted. For example, theWEEK interval are counted by Sundays rather than multiples of 7 days from the 'from date'. YEAR Intervals are counted from January 1st, not in 365-day multiples. This means the difference between December 31, 1999 and January 1, 2000 equals 1, the difference between January 1, 2000 and December 31, 2000 is 0. I call INTCK "the interval check function". The second function is INTNX which I remember as "the interval next function".

INTNX ('interval',start-from, increment<, 'alignment'>)
generates a SAS date, time, or datetime corresponding to the number of an interval from a start date, time or datetime value. The interval can be positive or negative.

Here is an example for both functions:

```
data interval;
        months= intck ('month',today(),'04jul2000'd ) ;
        last_month= intnx ('month',today(), -1) ;
        cur_month= intnx('month',today(), 0) ;
        next_month= intnx ('month',today(), 1) ;
```

```
    format  last_month cur_month next_month mmddyy10. ;
 put _all_ ;
 run ;
```

```
months=6   last_month=12/01/1999  cur_month=01/01/2000
 next_month=02/01/2000
```

NEW FEATURES Provided in Release 6.07 and 6.11

For Release 6.07, SAS Institute implemented 4 new date and datetime intervals, (WEEKDAY, TENDAY, SEMIMONTH, SEMIYEAR). Additionally you can code a parameter with the interval to shift the start point of the time interval. This feature is fully discussed in SUGI 25Coders Corner Paper 84-25 entitled "What Fiscal Year is this and when does it start and end?"
In Release 6.11 an optional parameter to adjust the alignment of the resultant date from INTNX has been added. Now you can get the beginning date, the middle, or the end of an interval period by passing a fourth argument. BEGINNING is the default value. The MIDDLE parameter returns the period's midpoint date and the END argument passes the end of the period date. " 'b','m','e' " can be used a abbreviation.
Here is an example:

```
data _null_ ;
    startmonth = intnx('month',today(),-1, 'b');
    middle = intnx('month',today(),-1,'m');
    endmonth = intnx('month',today(),-1,'e');
    format startmonth middle endmonth date. ;
    put _all_ ;
  run ;
```

```
startmonth=01DEC99  middle=16DEC99  endmonth=31DEC99
```

Using Dates/Times in IF and WHERE statements

Many times you will want to select records from a data set based on a time criteria. For example: you want to print all the problems occurring last month from your company problem management system. This is easily done by coding a subsetting if statement:

```
    If '01dec1999' d  <=  occured_date  <=  '31dec1999' d ;
```

“WHERE” statements can be used in both data steps and procedures. This next example gets all the problem records which were closed in year 1998:

```
    where year(closed_date) = 1998 ;
```

A compound statement could be used to get the last quarter of 1998 from a datetime variable:

```
     where year(datepart(closed_dt ))= 1998 and
qtr(datepart(closed_dt))= 4 ;
```

Puting it all Together With PROC's and WHERE Statements

Once you have created your SAS DATES in data sets, you will want to build output reports and graphs. Date Formats are very powerful when creating reports and graphs from SAS data sets. You can use date formats to aggregate data from a lower to a higher period (i.e. daily to monthly or monthly to yearly). This is easily done in SAS procedures such as FREQ, SUMMARY, and GRAPH without coding extra data steps . The following is an example using PROC FREQ:

```
proc freq data= sprod.prb1998 ;
     where year(occdate)=1998 and
     qtr(occdate) = 1 ;
     tables occdate/list nocum ;
     format occdate monyy7. ;
run ;
```

Executing this code results in this table:

Occurred Date

OCCDATE	Frequency	Percent
JAN1998	3500	32.4
FEB1998	3466	32.1
MAR1998	3846	35.6

Notice I used the format MONYY7. I code this format frequently. It is extremely useful for preparing summary reports and graphs. SAS software aggregated the daily occured dates to monthly counts. I did not have to code a data step to change daily values to a month value.

Post Y2K Issues

Now that we are past January 1, 2000 you might think we no longer have to be concerned with Y2k. Well no, there are plenty of legacy SAS reporting programs which are only run periodically and have not executed since 1999. If your shop is like mine, I expect you will find some infrequently run programs no longer work when next executed. As SAS programmers we still must be concerned with external files containing 2 digits dates. We still must look for embedded '19' in titles and macro variables. And be careful using the MDY function when the code has computed the year. We had a case where the year value was 3 digit number. This occurred when '1900' was subtracted from a 4 digit year resulting in a 2 digit year value from 1900 to 1999 but computed 100 for the year 2000. MDY does not work with 3 digit years. It only computes with a 2 digit or 4 digit. Again I refer you to Paper 84-25 in Coders Corner section.

YEARCUTOFF Option

Finally a discussion on how to use the YEARCUTOFF system option is in order. The YEARCUTOFF option allows you to create a 100 year window using SAS software. The default for Version 7 and above is 1920. This means when the SAS system has to convert a 2 digit year if the value is less then 20 it is consider to be in the 21st century and If the year is greater or equal to 20 then the year is in the 20th century. Windowing works as long the date values do not span over 100 years. It depends on the data. Accounting dates will most likely work but a Birthday value could not work if the data set of people contains elderly, middle age, and babies. The best approach there is to have 4 digit years. Here is a YEARCUTOFF example:

```
options yearcutoff=1950;
        data testdate;
            input @01 name $10.
                  @12 birthdate mmddyy6.
                  @20 startdate yymmdd10.;
            startage=int((startdate-birthdate)/365.25);
            agenow  =int((date()-birthdate)/365.25);
            format birthdate startdate monyy7. ;
datalines;
Gretchen   120574  1997/12/23
Mike       092547  1966/06/15
Wayne      092040  1966/08/01
run;
proc print data=testdate;
run;
```

The output looks not correct :

name	birthdate	startdate	startage	agenow
Gretchen	DEC1974	DEC1997	23	25
Mike	SEP2047	JUN1966	-81	-47
Wayne	SEP2040	AUG1966	-74	-40

The YEARCUTOFF equal to 1950 caused Mike and Wayne birthdays to be in the future rather than the past. And their ages were negative years(a virtual fountain of youth) . Always make sure you know what the YEARCUTOFF options are set to. (Please refer to the example under the heading "Arithmetic Date/Time Calculations" for the correct results)

Conclusion

The SAS system provides a complete set of DATE and TIME handling methods. I do not know of any other software having this amount of date/time products. I hope you the reader after studying this tutorial will be able to leverage these tools and methods. May your SAS coding be easier to write, be easier to maintain, and fun to use.

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

Wayne Finley,
"What Fiscal Year is this and when does it start and end? "
Proceedings of the Twenty-Fifth Annual SAS users Group International Conference, Cary, NC: SAS Institute, 2000

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

Year 2000 Processing with SAS® Software
Destiny Corporation
International SAS® Training and Consulting
100 Great Meadow Rd. Suite 601
Wethersfield, CT 06109-2355

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

SAS Institute Inc., SAS Language and Procedures: Usage, Version 6, First Edition, Cary, NC: SAS Institute, Inc., 1989

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

SAS Institute, Inc., SAS® Software: Changes and Enhancements, Release 6.11, Cary, NC: SAS Institute, Inc., 1995

Acknowledgments

I would like to acknowledge Andrew H. Karp, Dana Rafiee, and Susan Slaughter for all of their encouragement and support.

Contact Information

I welcome any comments or questions. I can be contacted at:

Wayne Finley
Health & Human Services 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