How to Read, Write, and Manipulate SAS® Dates

Jenine Milum, Charlotte, NC

ABSTRACT

No matter how long you’ve been programming in SAS, using and manipulating dates still seems to require effort. Learn all about SAS dates, the different ways they can be presented, and how to make them useful. This paper includes excellent examples in dealing with raw input dates, functions to manage dates, and outputting SAS dates into other formats. Included is all the date information you will need: date and time functions, Informats, formats, and arithmetic operations.

WHAT IS A SAS DATE?

A date is unique within SAS programming. It is neither a character value nor a typical numeric. It is a special case of a numeric variable. In other words, a SAS date is a special representation of a calendar date.

Unlike dates in many other languages, SAS has a specific numeric value assigned to each day. The starting point for all SAS dates is January 1st, 1960 and is represented as day zero (0). All previous and subsequent dates are represented with numeric values plus (+) or minus (-) from this starting point. The simplicity of the approach is there will never be a point in the past (since the start of the Gregorian calendar) or future that can not be represented by a number.

December 30, 1959	←→	-2
December 31, 1959	←→	-1
January 1, 1960	←→	0
January 2, 1960	←→	1
January 3, 1960	←→	2

January 23, 1963	←→	1118
Oct 20, 2008	←→	17825

How to Test a Date

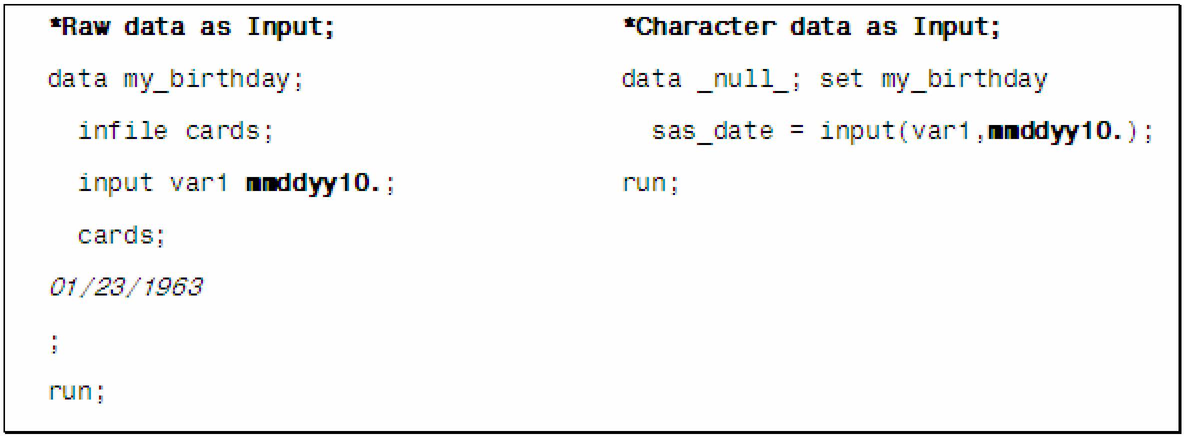
There will be times you may want to test a date to see what date we know and its corresponding SAS date.

```
Data _null_;
  Today = date();
  Testdate = ’23jan63’d;
  firstdate = ’01jan1960’d;
Put ‘Log shows: ‘ today testdate firstdate;
Run;
Log shows: 17825 1118 0
```

Assigning and setting a date is done by one of the 2 methods above. Either a 2 digit year or a 4 digit year is acceptable. The main point is the quote marks followed immediately by a **d**. This notation allows SAS to know the contents inside the quotes are a **d**ate.

CONVERTING A TEXT DATE TO SAS DATE - INFORMATS

Often the dates we'll be using in our programming comes to us in the form of a character value from a text field. There are several ways to let SAS know the value is really a date and allow SAS to convert it into true SAS date. Utilizing SAS **informats** is the method that is used.



```
*Raw data as Input;

data my_birthday;

  infile cards;

  input var1 mmddyy10.;

  cards;

01/23/1963

;

run;
```

```
*Character data as Input;

data _null_; set my_birthday

  sas_date = input(var1,mmddyy10.);

run;
```

These examples work for all versions of SAS. It takes the information from text and tells SAS the information is a date in a month-month, day-day, year-year format for a length of 10. SAS assumes the slashes (/), or other acceptable separation character, is included in the length of 10.

SAS INFORMATS

Below are some of the more common SAS date Informats and the text values they will convert. These informats are valid for all SAS versions.

INPUT		INFORMAT
01/23/1963	←→	mmddyy10.
1/23/1963	←→	mmddyy10.
01/23/63	←→	mmddyy8.
1/23/63	←→	mmddyy8.
January 23,1963	←→	worddate20.
jan 23, 1963	←→	worddate12.
23jan1963	←→	date9.
23jan63	←→	date7.
23-jan-1963	←→	date11.
01-23-63	←→	mmddyy8.
19630123	←→	yymmdd8.

In V9 of SAS, all the above date text values can be converted to SAS utilizing just one Informat**: *anydtdte15.***

SYSTEM DATES

There are many situations where a date needs to be dynamic. SAS provides several opportunities to extract a date from the system you are running. Below are a few of the system dates.

```
data _null_;
  a = date();
  b = today();
  c = "&sysdate"d;
  d = "&sysdate9"d;
put ‘Log shows: ‘ a b c d;
run;


Log shows: 17825 17825 17825 17825
```

As you can see, each returns the same results. Although, **&sysdate9** returns the date the session started, not necessarily the current date. The system dates are treated as macro variables and require residing between double quotes with the **d** designation. Now you have a variable with a valid SAS date.

Date Functions

Because date variables are unique within SAS, they have their own set of functions. It may be you want to extract something specific from a date while still maintaining it being a SAS date value. Such a situation would be if you were interested in what day of the week a specific date occurred on. Below are a handful of date functions and what results from utilizing them would produce.

```
data _null_;
     my_birthday = '23jan63'd;
     date1 = day(my_birthday);
    date2 = month(my_birthday);
    date3 = year(my_birthday);
    date4 = qtr(my_birthday);
     date5 = weekday(my_birthday);
    put date1 date2 date3 date4 date5;
run;


Log shows:
23 1 1963 1 4
```

As you can see from the example above, the **day function** returns the day of the month, **month function** returns the month of the year and the **year function** returns the 4 digit year. The **qtr function** returns the quarter of the year where January - March are quarter 1, etc. The **weekday function** returns values 1 through 7 where day 1 is Sunday and so on representing the 7 days of the week.

Date Functions and Date Arithmetic

Being able to determine additional information from dates also is endeavored using special SAS date functions. You may use them to determine the time between two intervals, alter a point in time by a duration of time, and subtract time intervals. Below are a few of them.

```
data _null_;
  date = date();
  my_birthday = '23jan63'd;
  datedif2 = intck('month',my_birthday,date);     * intck('interval', from, to );
  datedif3 = sum(date,-my_birthday);              * sum(to,-from);
  datedif1 = datdif(my_birthday,date,'act/act');  *  datdif(from,to,'act/act')
                                                            or '30/360';
  put 'Log shows: ' datedif1 datedif2 datedif3;
run;


Log shows: 16707 549 16707
```

The **intck function** returns the number of "intervals" from one date to another. I this case, with the interval of Month, it is the number of months from Nov 5th, 2007 to my birth.

The **sum function**, while very valid with other forms of numeric's, also works with dates. As expected, it returns the number of days between one date interval and another.

The **datdif function,** in this example using 'act/act' (actual/actual), returns the number of days between one point in time and another point in time. The result here is the number of days from my birth to Nov 5th. 2007.

The **intck function** can be a very handy function while working on dates. Looking into this function further, below is a handful of "intervals" and their returned results.

```
data _null_;
  my_birthday = '23jan1963'd;
    years = intck('year',my_birthday,date());
 quarters = intck('qtr',my_birthday,date());
   months = intck('month',my_birthday,date());
    weeks = intck('week',my_birthday,date());
     days = intck('day',my_birthday,date());
 put 'Log shows: ' years quarters months weeks days;
run;


Log shows: 45 183 549 2387 16707
```

The results from the previous data step shows, in order, how many years, how many quarters, how many months, how many weeks and how many days old I am. All of this with using just the one date function and altering the "interval".

Date Formats

Outputting dates from SAS is yet another ability of SAS to manipulate dates. Most people don't know what today's date is in SAS. People need to be able to visualize what date is represented in one of the normal and acceptable forms. Date formats are used to control the look and results of dates that are currently in SAS form. You can presents dates in data fields, in report titles or labels. There are quite a few date formats. They may be easily located in SAS Help and other SAS Documentation. Below are a few to give you a feel of how to use them.

```
data dates;
   my_birthday = '23jan1963'd;              * SAS date 1118;
   date1 = put(my_birthday,mmddyy8.);
   date2 = put(my_birthday,worddate15.);
   date3 = put(my_birthday,monyy7.);
   date4 = put(my_birthday,julian5.);
   put date1  date2  date3 date4;
run;


Log Shows:
01/23/63     Jan 23, 1963     JAN1963      63023
```

The formats above clearly control the look of a SAS date. Even though the date being used above is the SAS date 1118, it is output in a form familiar to us all.

Warning signs of bad dates

There are always times when SAS dates aren't manipulated properly. Signs of such occasions are null values (.) or the occurrence of too many January 1st, 1960 values, meaning the date really returned a zero.

A listing of most SAS Date Functions, Informats and Formats – All in one place!

Date and Time Functions

DIF returns the number of days between two dates

DATE returns the current date as a SAS date value

DATEJUL converts a Julian date to a SAS date value DATEPART extracts the date from a SAS datetime value

DATETIME returns the current date and time of day as a SAS datetime value

DAY returns the day of the month from a SAS date value

DHMS returns a SAS datetime value from date, hour, minute, and seconds

HMS returns a SAS time value from hour, minute, and seconds

HOUR returns the hour from a SAS time or datetime value

INTCK returns the integer number of time intervals in a given time span

INTNX advances a date, time, or datetime value by a given interval, and returns a date, time, or datetime valueJUL

DATE returns the Julian date from a SAS date value

JULDATE7 returns a seven-digit Julian date from a SAS date value

MDY returns a SAS date value from month, day, and year values

MINUTE returns the minute from a SAS time or datetime value

MONTH returns the month from a SAS date value

QTR returns the quarter of the year from a SAS date value

SECOND returns the second from a SAS time or datetime value

TIME returns the current time of dayTIMEPARTextracts a time value from a SAS datetime value

TODAY returns the current date as a SAS date valueWEEKDAYreturns the day of the week from a SAS date value

WEEKDAY returns an integer that represents the day of the week, where 1=Sunday, 2=Monday,⊤, 7=Saturday

YEAR returns the year from a SAS date valueYRDIFreturns the difference in years between two dates

YYQ returns a SAS date value from the year and quarter

Date Informats

DATE*w.* day, month abbreviation, and year:

17oct91 *ddMONyy*

DATETIME*w.d* date and time: *ddMONyy:hh:mm:ss*

17oct91:14:45:32

DDMMYY*w.* day, month, year: *ddmmyy*, *dd/mm/yy*,

17/10/91 *dd-mm-yy*, or *dd mm yy*

JULIAN*w.* year and day of year (Julian dates): *yyddd*

91290

MMDDYY*w.* month, day, year: *mmddyy*, *mm/dd/yy*,

10/17/91 *mm-dd-yy*, or *mm dd yy*

MONYY*w.* month abbreviation and year

Oct91

TIME*w.* *d*hours, minutes, seconds: *hh:mm:ss*

14:45:32or hours, minutes: *hh:mm*.

YYMMDD*w.* year, month, day: *yymmdd*, *yy/mm/dd*,

91/10/17 *yy-mm-dd*, or *yy mm dd*

YYQ*w.* year and quarter of year: *yyQq*

91Q4

Date Formats

DATE*w.* day,month abbreviation,year: d*dMONyy*

17oct91

DAY*w.* day of month

17

DDMMYY*w.* day,month,year: d*d/mm/yy*

17/10/91

DOWNAME*w.* name of day of the week

Thursday

JULDAY*w.* day of year

290

JULIAN*w.* year and day of year: *yyddd*

91290

MMDDYY*w.* month, day, year: *mm/dd/yy*

10/17/91

MMYY*w.* month and year: *mmMyy*

10M1991

MMYYC*w.* month and year: *mm:yy*

10:1991

MMYYD*w.* month and year: *mm-yy*

10-1991

MMYYP*w.* month and year: *mm.yy*

10.1991

MMYYS*w.* month and year: *mm/yy*

10/1991

MMYYN*w.* month and year: *mmyy*

101991

MONNAME*w.* name of month

October

MONTH*w.* month of year

10

MONYY*w.* month abbreviation and year:

OCT91

MONyy*QTR*w. quarter of year

4

QTR*w.* quarter of year

4

QTRR*w.* quarter in Roman numerals

IV

WEEKDATE*w.* *day-of-week, month-name dd,yy*

Thursday, October 17, 1991

WEEKDATX*w.* *day-of-week, dd month-name yy*

Thursday, 17 October 1991

WEEKDAY*w.* day of week

5

WORDDATE*w.* *month-name dd, yy*

October 17, 1991

WORDDATX*w.* *dd month-name yy*

17 October 1991

YEAR*w.* year

1991

YYMM*w.* year and month: *yyMmm*

1991M10

YYMMC*w.* year and month: *yy:mm*

1991:10

YYMMD*w.* year and month: *yy- mm*

1991-10

YYMMP*w.* year and month: *yy.mm*

1991.10

YEAR*w.* year

1991

YYMM*w.* year and month: *yyMmm*

1991M10

YYMMC*w.* year and month: *yy:mm*

1991:10

YYMMD*w.* year and month: *yy- mm*

1991-10

YYMMP*w.* year and month: *yy.mm*

1991.10

YYMMS*w.* year and month: *yy/mm*

1991/10

YYMMN*w.* year and month: *yymm*

199110

YYMON*w.* year and month abbreviation:

1991OCT

YYMMDD*w.* year, month, day: y*y/mm/dd*

91/10/17

YYQ*w.* year and quarter: *yyQq*

91Q4

YYQC*w.* year and quarter: *yy:q*

1991:4

YYQD*w.* year and quarter: *yy-q*

1991-4

YYQP*w.* year and quarter: *yy.q*

1991.4

YYQS*w.* year and quarter: *yy/q*

1991/4

YYQN*w.* year and quarter: *yyq*

19914

YQR*w.* Yyear and quarter in Roman

1991

QIVnumerals: *yyQrr*

YYQRC*w.* year and quarter in Roman

1991:IV numerals: *yy:rr*

YYQRD*w.* year and quarter in Roman

1991-IV numerals: *yy-rr*

YYQRP*w.* year and quarter in Roman

1991.IV numerals: *yy.rr*

YYQRS*w.* year and quarter in Roman

1991/IV numerals: *yy/rr*

YYQRN*w.* year and quarter in Roman

1991IV numerals: *yyrr*

CONCLUSION

Presented here are the most common tools used while working with dates using SAS. They are part of nearly all programming efforts in one capacity or another. Quite frequently, there is more than one valid method to working with dates. Understanding dates is the best way to learn how to manipulate them.

REFERENCES

SAS 9.1.3 Help and Documentation “Your Complete Guide to Syntax, How To, Examples, Procedures, Concepts, What’s New, and Tutorials.”

ACKNOWLEDGMENTS

A big “Thank you” to several friends that helped me prepare this paper and presentation. A special thanks to my friends in Cameroon and Kenya for allowing me to practice this presentation in their attendance.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jenine Milum
4013 Kingsgate Place
Charlotte, North Carolina
E-mail: JenineMi@yahoo.com