How to Handle a Hot Date
(Fancy and not so Fancy Date Manipulation with SAS)

David Beam
Systems Seminar Consultants
Madison, WI 53716 (608) 222-7081

Abstract
One of the areas that SAS Software excels is the powerful capabilities to do sequential date processing. This paper will present a brief overview of how SAS handles dates, and some examples of conversions required to read in a variety of dates.

Introduction
The SAS method for doing date comparisons, math on date fields, sorting dates etc. is to use a numeric sequential representation of a date value. All date values in SAS are converted in SAS to a numeric value representing the number of days from Jan. 1, 1960. Times of day can also be represented as a numeric value inside SAS, converted to the number of seconds that has passed since midnight. A combination of a date and a time of day can also be used in SAS, internally the value is the number of seconds from midnight on January 1, 1960.

We use a variety of *informs* and *formats* to read in these dates and display them correctly, and a wide variety of *functions* are available to manipulate the dates as needed.

Sample of some dates/times in SAS:
What we see: What SAS has:
01/27/1996 13,175
01/27/1954 -2,165
06:45 PM 67,500
06:45PM, 1/27/1954 -186,988,500

A Brief List of some SAS Informats/Formats
This is by no means meant to be a comprehensive list, refer to the SAS documentation for complete lists:
Format/Informat: Example:
MMDDYY10. 02/15/1996
MMDDYY6. 021596
YYMDD10. 1996-02-15
DATE9. 15FEB1996
JULIAN5. 96046

A Brief List of some SAS date/time Functions
Again, this is a partial list of some of the SAS functions used to manipulate dates and times of day:
Function: Description:
DATE Returns current date
DAY Returns the day of month from a SAS date
INTCK Return number of intervals between dates
INTNX Advance date/time by an interval
MDY . Return a SAS date from month, day, year

Reading in a few dates:
The following examples show how to read in a few dates from a non-SAS data file by using a combination of SAS inormats and functions.

Example 1:
Reading a DB2 style of date, coming in as 1996-02-15:
  INPUT @1 bdate YYMDD10.
Results in a numeric value 13,194.

Example 2:
Suppose you have a COBOL file, with a date stored as Julian but "packed decimal". This requires two SAS statements, to read a Julian date and then to convert:
  INPUT @1 bdate PD3.; /* result is value 96046 */
  bdate = DATEJUL(bdate);
  /* result is value 13194 */

Example 3:
Now suppose the date is a numeric value, but the value in the input field contains month, day and year as 2151996, for Feb. 15, 1996.
This must be read in as a numeric value and converted to a SAS date value by first creating a character string and passing that through the correct date informat with the INPUT function:
  INPUT @1 bdate PD5.; /* result: 2151996 */
  bdate = INPUT((PUT(bdate),$8.), MMDDYY8.); /* results in SAS date value */

Example 4:
In the worst case, if you can't use any informat or function directly to convert your date to a SAS date value, try to get the month, day and year of the date coming in as a numeric value. Then the MDY function can be used to create the SAS date value:
  INPUT @1 month 2. /* month = 2 */
  @65 year 4. /* year = 1996 */
  @97 day 2.; /* day = 15 */
  BDATE = MDY(month, day, year);
  /* results is value of BDATE = 13194 */

The Year 2000 - Be Careful!
Coders' Corner

If you are working with data that only has a two digit year, as in "02/15/37", you need to be aware that SAS will only see the year as 37, and SAS needs to make an assumption as to whether this is 1937 or 2037.

A SAS global option called "YEARCUTOFF" is available to control this situation. The YEARCUTOFF= option tells SAS to use the current century or the next century when working with only a two digit year.

If the two digit year is LESS than the last two digits of the YEARCUTOFF= setting, SAS assumes the NEXT century. If the two digit year is the SAME or GREATER than the last two digits of YEARCUTOFF=, SAS assumes the same century as YEARCUTOFF=.

For example, let's assume:

```
OPTIONS YEARCUTOFF=1950;
```

Your two digit years will be interpreted as follows:

<table>
<thead>
<tr>
<th>Input Date</th>
<th>SAS Translates to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>01JAN49</td>
<td>January 1, 2049</td>
</tr>
<tr>
<td>01JAN50</td>
<td>January 1, 1950</td>
</tr>
<tr>
<td>01JAN51</td>
<td>January 1, 1951</td>
</tr>
<tr>
<td>01JAN05</td>
<td>January 1, 2005</td>
</tr>
</tbody>
</table>

The default from SAS is...

```
OPTIONS YEARCUTOFF=1900;
```

... which results in ALL two digit years falling into the 1900s.

Hard Coding SAS Dates or Times

Often in a SAS program you will need to type in a value of a date, time or datetime for comparison or computing with dates. The Date, Time and Date/Time constants as shown below allow 'hard-coding' of dates and times:

```
'15FEB1996'D   == date constant
'14:45'T       == time constant
'15FEB1996:14:45DT == Date/Time constant
```

Now the following code can be used, for example, to select all invoices between two date ranges.

```
WHERE inv_dte GE '15FEB1996'D
AND inv_dte LE '29FEB1996'D;
```

Using Simple SAS Date Functions

Most of the SAS date/time functions are easy to use and understand. The following functions all return a numeric value when a SAS date value is passed as the argument (with 15FEB1996 as the current date):

```
Today = DATE();   /* get today's date */
Year = YEAR(today); /* returns: 1996 */
Quarter = QTR(today); /* returns: 1 */
Month = MONTH(today); /* returns: 2 */
Day = DAY(today); /* returns: 15 */
```

Interval Functions

Two very special date/time functions need to be discussed. These are used to compute a date/time in the future or past, or compute how many of a specified interval passed between two dates or times.

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

Returns the number of time intervals in a given span. The count is how many of the specified interval began between the dates.

```
INTNX('interval', from, number, 'alignment')
```

Advances a date/time value by the specified value, to the date/time at the requested alignment. If no alignment given, the date/time is advanced to the beginning of the specified interval. (The "alignment" argument is new with release 6.11 and will cause an error if used in earlier versions of SAS).

In both of the above, the interval argument is a character string representing the requested interval:

```
Date intervals         Time intervals       Datetime intervals
DAY                   HOUR                    DT\$DAY
WEEK                  MINUTE                 DT\$WEEK
MONTH                 SECOND                 DT\$MONTH
QTR                  SECOND                 DT\$QTR
YEAR                  SECOND                 DT\$YEAR
```

The following intervals were added with release 6.07, for both Date and DateTime intervals:

```
WEEKDAY                TENDAY                SEMIMONTH
SEMICYEAR              SEMICYCLE             SEMINYEAR
```

The new 'alignment' argument specifies the interval to advance to. The following values are available:

```
BEGINNING' or 'B'      beginning of the interval
(midpoint of the interval)

'MIDDLE' or 'M'        midpoint of the interval

'END' or 'E'           end of the interval
```

Example 1

Using the INTCK function with NOW = '15FEB1996'D:

```
INTCK('WEEK', now, '19FEB1996'D);
returns 1
```

```
INTCK('MONTH', now, '23APR1996'D);
returns 2
```

```
INTCK('MONTH', now, '19FEB1996'D);
returns 0
```

```
INTCK('TENDAY', now, '29FEB1996'D);
returns 1
```

```
INTCK('SEMICYEAR', now, '31DEC1996'D);
returns 1
```

Example 2

Using the INTNX function with NOW = '15FEB1996'D:

```
INTNX('WEEK', now, 2);
returns 25FEB1996
```
INTNX('WEEK', now, 1, 'E');
    returns 24FEB1996
INTNX('WEEK', now, 1, 'M');
    returns 21FEB1996
INTNX('MONTH', now, 0);
    returns 01FEB1996
INTNX('MONTH', now, 0, 'END');
    returns 29FEB1996

How to Get Today's Date in a Title
There is a SAS system macro variable, &SYSDATE, that will return today's date as a character string, in the SAS DATEDT7. format. For example:

    TITLE "This was run on &SYSDATE";
will resolve to a title:

    This was run on 15FEB96

NOTE: Be sure to use double quotes in the TITLE, the &sysdate will not resolve if single quotes are used.

What if you don't like the style SAS uses for this date?
Simple...create your own macro variable:

DATA _null_;
    /* get today's date as character string */
    ch_date = "&SYSDATE";
    /* convert to numeric date value */
    nu_date = INPUT(ch_date, DATE7.);
    /* put to character string in desired date format */
    my_date = PUT(nu_date, MMDDYY10.);
    /* create MACRO variable holding formatted date */
    CALL SYMPUT("mydate", my_date);
run;
title "New date is &MYDATE";
    resolves to:  New Date is 02/15/1996

Real-World Example
The following shows where date manipulation in SAS can really help. Suppose your department is responsible for tracking the performance of vendors. The boss wants to know how long it takes, in hours, for a vendor to respond to a question. You must ignore the hours from 5:00pm to 8:00am the next day, as the office is closed. You must also ignore the weekends.

If you record the date and time when you call the vendor, and the date and time when they call you back, you can subtract the two values and compute the response time in hours. Use functions to check if an overnight must be deducted, and to count the weekends that have passed:

(In the following program the TIME variable is computed as MINUTES, so is converted to hours before printing.)

DATA vendors;
    INPUT @1 called   datetime13.  @15 answer  datetime13.;
    time = answer - called;
    /* see if overnight, DATEPART gets date alone */
    IF DATEPART(answer) GT DATEPART(called) then DO;
    /* remove 5-12PM and 12PM-8AM hours */
    /* or 15 hours per day */
    days = INTCK('DTDAY', called, answer);
    time = time - (15 * 60 * 60 * days);
    END;
    /* check for weekends, remove an extra 9 hours */
    /* for each of the two weekend days */
    weeks = INTCK('DTWEEK', called, answer);
    IF weeks > 0 then DO;
    /* add back the hours lost each weekend */
    time = time - (9 * 2 * 60 * 60 * weeks);
    END;
    hours = (time / 60) / 60;
    hours = ROUND(hours,.01);
    DROP days time weeks;
    FORMAT called answer DATETIME13.;
    DATALINES;
15FEB96:03:45 15FEB96:14:22
15FEB96:04:45 16FEB96:14:22
15FEB96:08:45 05MAR96:08:10
;RUN;

and the final results:

    CALLED       ANSWER       HOURS
15FEB96:08:45 15FEB96:14:22  5.62
15FEB96:08:45 16FEB96:14:22  14.62
15FEB96:08:45 05MAR96:08:10 116.42

CONCLUSION
Understanding dates and times and how SAS handles them can help solve many business analysis problems. It is easy to set up a sample test program as shown above, to test your calculations.

SAS is a registered trademark of SAS Institute Inc.

The author will be glad to answer questions and accept suggestions at the following address:

David Beam
Systems Seminar Consultants
1220 Fernite Drive
Suite 206
Madison, WI 53716

voice:    (608) 222-7081
Internet: train@sys-seminar.com