Paper 66-28

DATE HANDLING IN THE SAS ® SYSTEM

Bruce Gilsen, Federal Reserve Board

1. Introduction

A calendar date can be represented in the SAS ® system as a SAS date value, a special representation of a calendar date provided by SAS software. SAS date values are easy to use once a few basic concepts are understood.

This paper shows how to create, transform, and print SAS date values. It reviews tools used with SAS date values, such as formats, informats, and DATA step functions, and the effect of the SAS system option YEARCUTOFF.

The paper focuses on SAS date values, but also reviews other ways to represent calendar dates, and some problems that can occur when SAS date values are not used.

2. What is a SAS date value?

Simply put, a SAS date value is an integer whose value is the number of days before or after January 1, 1960. Here is how some dates are represented as SAS date values (don't worry yet about how SAS date values are created).

Calendar date	SAS date value
December 30, 1959	-2
December 31, 1959	-1
January 1, 1960	0
January 2, 1960	1
January 3, 1960	2
.....	.....
January 1, 2000	14610
.....	.....
September 1, 2001	15219

The SAS system can process SAS date values from A.D. 1582 to A.D. 19900 (though day of the week cannot be accurately determined for dates before September 1752).

3. How are SAS date values created?

Sections 3.1-3.4 illustrate the following ways to create a SAS date value.

1. Read raw data from a file with an informat.

2. Convert character or numeric variables to SAS date values with an informat or function.

3. Use a SAS date constant.

4. Retrieve the current date with &SYSDATE, &SYSDATE9, or DATE( ).

In sections 3.1-3.4, the SAS system option YEARCUTOFF (discussed in section 4, "Creating SAS date values: two- and four-digit years and the YEARCUTOFF option") is assumed to have the default value, 1920.

3.1. Read raw data from a file with an informat

An informat is an instruction that tells the SAS system how to read data values. The SAS system provides many informats to read raw data values that are calendar dates and transform them into SAS date values.

The appropriate informat depends on how the calendar dates are stored in the input file. For example, a different informat is needed to create a SAS date value from each of the following raw data values that represent June 2, 1960.

```
6/2/60
02jun60
06021960
6-2-60
```

In the following DATA step, the variables dat1 and dat2 are SAS date values created using the DATE7. and YYMMDD8. informats, respectively. For illustrative purposes, the data values include two representations of the same date, which is unlikely in a real application.

```
data one ;
     input     @1 dat1 date7.
               @9 dat2 yymmdd8.;
     datalines;
01jan60 19600101
02jan99 19990102
01jan00 20000101
; run;
```

Note that if the data are read from a file, the DATALINES statement and the data values would not be included, and an INFILE statement would be added to refer to the file.

Informats are documented in the *SAS Language Reference, Version 8*. A summary table by category on pages 640-643

provides a useful quick reference.

3.2. Convert character or numeric variables to SAS date values with an informat or function

In the first DATA step below, the following variables are created. They are typical examples of how calendar date data are represented before conversion to SAS date values.

dat1 is a SAS date value

mm, dd, and yy are numeric variables that contain the month, day, and year for a calendar date

char1 is a character variable of length 8

char2 is a character variable of length 7

num1 is a numeric variable

In the second DATA step below, variables containing date data are converted to SAS date values. The variables dat1-dat5 are SAS date values with the same value.

```
data one ;
    input     @1 dat1 date7.
              @9 mm 2.
              @12 dd 2.
              @15 yy 2.
              @18 char1 $8.
              @27 num1 8.
              @36 char2 $7. ;
    datalines;
01jan60 01 01 60 19600101 19600101 1960JAN
31dec59 12 31 59 19591231 19591231 1959DEC
02jan99 01 02 99 19990102 19990102 1999JAN
; run;

data two ;
    set one ;

    /* Without this statement, flipmon is created with
    the same length as char2, 7. */
    length flipmon $3 ;

    /* Create a SAS date value from month, day, and
    year values with the MDY function. */
    dat2 = mdy(mm, dd, yy) ;

    /* Create a SAS date value from a character
    variable with the INPUT function. The second
    argument to the INPUT function is an informat
    (YYMMDD8. in this example because that is how
    the data are formatted). */
    dat3 = input(char1, yymmdd8.) ;

    /* To create a SAS date value from a numeric
    variable, first convert the numeric variable to a
    character variable with the PUT function. Then,
    convert the character variable to a SAS date value
    with the INPUT function. */
    tempchar = put (num1, 8.) ;
    dat4 = input(tempchar, yymmdd8.) ;

    /* This statement is equivalent to the previous two
    statements. It avoids creating the variable
    tempchar, but is somewhat less readable. */
    dat5=input(put(num1,8.),yymmdd8.) ;

    /* If the data do not correspond to a date informat,
    transform the data and then create SAS date
    values. For example, no date informat is available
    for variable char2, of the form 1960JAN. Switch
    the month and year, then create SAS date values
    with the MONYY. informat. */
    flipmon=substr(char2, 5, 3);   /*extract month */
    flipyear = substr(char2, 1, 4);  /* extract year */
        /* concatenate giving mmmyyyy */
    flipwork = flipmon || flipyear;
    dat6=input(flipwork, monyy7.);  /* SAS date */

    /* This statement is equivalent to the previous four
    statements. It avoids creating 3 variables, but is
    less readable. */
    dat7 = input(substr(char2, 5, 3) ||
      substr(char2,1, 4), monyy7.);
```

3.3. Use a SAS date constant

A SAS date constant is a value of the form ddMMMyy or ddMMMyyyy in single or double quotes, followed by the letter D. It represents a single SAS date value.

The statements below create SAS date values for June 2, 1960 using a two-digit year (dat1, dat2, dat3) or a four-digit year (dat4, dat5, dat6). A four-digit year is preferred, because the YEARCUTOFF option determines the century of SAS date values created with a two-digit year (see section 4, "Creating SAS date values: two- and four-digit years and the YEARCUTOFF option").

```
dat1 = '02jun60'd ;
dat2 = "02jun60"d ;
dat3 = '02jun60'D ;
dat4 = '02jun1960'D ;
dat5 = "02jun1960"d ;
dat6 = "02jun1960"D ;
```

3.4. Retrieve the current date with &SYSDATE, &SYSDATE9, or DATE( )

&SYSDATE and &SYSDATE9 are automatic macro variables (macro variables provided by the SAS system) that contain the date your SAS session began.

&SYSDATE9, new in Version 8 of the SAS system, uses the format ddmmmyyyy (02jun1960).

&SYSDATE uses the format ddmmmyy (02jun60). Many existing applications use &SYSDATE.

If your SAS session began on September 1, 2001, the following statements set dat7 and dat9 to 15219, the SAS date value for September 1, 2001.

```
dat7 = "&SYSDATE"d ;
dat9 = "&SYSDATE9"d ;
```

&SYSDATE9, not &SYSDATE, is recommended. Since &SYSDATE has a two-digit year, the YEARCUTOFF option determines the century of dat7 but not dat9. The YEARCUTOFF option is described in the next section.

The DATA step function DATE creates a SAS date value whose value is today's date, as follows.

```
dat8 = date( ) ;
```

TODAY is an alias for DATE.

4. Creating SAS date values: two- and four-digit years and the YEARCUTOFF option

In the previous sections, SAS date values were created from raw data, SAS character or numeric variables, and date constants. When SAS date values are created, there is an important difference between input values with a two-digit year (e.g., 96) and those with a four-digit year (e.g., 1996).

If a SAS date value is created from a four-digit year, the calendar date used to create the SAS date value is unambiguous.

If a SAS date value is created from a two-digit year, the century is determined by the SAS system option YEARCUTOFF. YEARCUTOFF specifies the first year of a 100-year span used to determine the century of a SAS date created from a two-digit year.

In Version 8 of the SAS system, the value of YEARCUTOFF and the corresponding range of SAS date values created from a two-digit year are as follows.

SAS Institute sets YEARCUTOFF by default to 1920. SAS date values created from a two-digit year have values between January 1, 1920 and December 31, 2019.

YEARCUTOFF can be customized at your site. For example, at the Federal Reserve Board, the default value of YEARCUTOFF is set to 1950. SAS date values created from a two-digit year have values between January 1, 1950 and December 31, 2049.

Some details about YEARCUTOFF include the following.

1. To reiterate, SAS date values created from four-digit years do not use the YEARCUTOFF option.

2. YEARCUTOFF does not affect dates that are not stored as SAS date values. Some users store date information as numeric or character variables instead of SAS date values.

3. SAS date values outside the YEARCUTOFF range must be created with a four-digit year.

4. The century of a SAS date value created from a two-digit year is determined by the YEARCUTOFF value in effect when the SAS date value is created. Subsequent changes to YEARCUTOFF do not affect existing SAS date values.

5. The following SAS statement displays the current value of YEARCUTOFF. The output is written to the SAS log.

```
proc options option = yearcutoff ;
```

6. You can set the YEARCUTOFF option in your application with an OPTIONS statement or as an invocation option when you invoke the SAS system. The following statement sets the YEARCUTOFF value to 1925.

```
options yearcutoff = 1925 ;
```

7. In Version 6, the default value of YEARCUTOFF was 1900. The default value could change in future versions. When converting to a new version of SAS software, check if YEARCUTOFF has changed. If it has, determine if the new value is appropriate for your site or application and if your application needs any modification.

5. Using SAS date values

This section shows how to transform, sort, and print SAS date values.

5.1. DATA step functions

The following DATA step functions can be used with SAS date values. These functions are documented in the *SAS Language Reference, Version 8*.

Function	Description
datdif	Returns the number of days between two SAS date values. New in Version 8.
date	Returns the current date as a SAS date value.

DATE and TODAY are aliases.

datejul — Converts a Julian date (number of the form yyddd or yyyyddd, where ddd is the day of the year) to a SAS date value.

datepart — Extracts the date from a SAS datetime value.

day — Returns the day of the month (1 - 31) of a SAS date value.

input — Creates a SAS date value from a character variable using an informat. The INPUT function is also used for data conversions unrelated to SAS date values.

intck — Returns the number of time intervals (days, weeks, months, quarters, or years) between two SAS date values.

intnx — Returns a SAS date value modified by a specified number of intervals (days, weeks, months, quarters, years, etc.). The new date is aligned to the beginning (the default), middle, or end of the interval, generating un-intuitive results for intervals other than day. See the examples in the next section.

juldate — Converts a SAS date value to a Julian date that has 5 digits if the date is in the YEARCUTOFF range, and 7 digits otherwise. The JULDATE7 function always returns a seven-digit Julian date and is recommended instead of JULDATE.

juldate7 — Returns a seven-digit Julian date from a SAS date value. New in Version 8.

mdy — Creates a SAS date value from month, day, and year values.

month — Returns the month (1 - 12) of a SAS date value.

put — Creates a SAS date value from a numeric variable using a format. The PUT function is also used for data conversions unrelated to SAS date values.

qtr — Returns the quarter of the year (1 - 4) of a SAS date value.

today — Returns the current date as a SAS date value. DATE and TODAY are aliases.

weekday — Returns the day of the week of a SAS date value as a number (1 = Sunday, 2 = Monday,..., 7 = Saturday).

year — Returns the four-digit year of a SAS date value.

yrdif — Returns the difference in years between two SAS date values. New in Version 8. Not likely to be useful, because it returns fractional values [yrdif('01may2000'd,'01jun2001'd,'actual')= 1.0830975372]. Use the INTCK function instead.

yyq — Returns the SAS date value of the first day of a quarter for a given year and quarter.

5.2. Calculations and functions in the DATA step

Since SAS date values are numeric values, they can be used in calculations. This is shown in the following DATA step, along with some of the DATA step functions summarized in the previous section. Section 3.2, "Convert character or numeric variables to SAS date values with an informat or function," includes examples of two other DATA step functions used with SAS date values, PUT and INPUT.

```
data _null_ ;

    /* today1 and today2 are set to a SAS date value
    whose value is today's date. */
    today1 = today( ) ;
    today2 = date( ) ;

    /* date1 = 13503, the SAS date value for
    December 20, 1996. */
    date1 = mdy(12, 20, 1996) ;

    /* Extract the month, day, quarter, year, and day of
    the week of the SAS date value date1
    (12/20/1996). */
    mon1 = month(date1) ;      /* mon1 = 12. */
    day1 = day(date1) ;        /* day1 = 20. */
    quarter1 = qtr(date1) ;    /* quarter1 = 4. */
    year1 = year(date1) ;      /* year1 = 1996. */
    /* wkday1=6 because 12/20/96 is a Friday. */
    wkday1= weekday(date1) ;

    /* date2 = 13057, the SAS date value for October
    1, 1995. YYQ always returns the SAS date value
    for the first day of the quarter. */
    date2 = yyq(1995, 4) ;

    /* date3 = 13496, the SAS date value for
    December 13, 1996. */
    date3 = date1 - 7 ;

    /* date4 = 13137, the SAS date value for
    December 20, 1995 (since 1996 is a leap year). */
    date4 = date1 - 366 ;

    /* date5 = 13496, the SAS date value for
```

```
December 13, 1996. */
date5=intnx('day', date1, -7) ;

/* date6 = 13380, the SAS date value for January
1, 1998, not December 20, 1998 as expected.
Why? By default, INTNX calculates from the
start of an interval. Interval is 'year', so INTNX
adds 2 years to January 1, 1996, the start of the
year. INTNX can calculate from the beginning,
middle, or end of the interval.*/
offset = 2;
date6=intnx('year', date1, offset) ;

/* date7=14233, the SAS date value for December
20, 1998, 2 years later than date1. This code adds
or subtracts years, accounting for leap years. See
Whitlock (1999) for a generalized macro that
changes SAS date values by days, weeks, months,
or years. */
offset = 2;
if day(date1) = 29 and month(date1) =2 and
   mod(offset,4) ne 0
then date7 = mdy(2, 28, year(date1)+offset) ;
else date7 = mdy(month(date1),
   day(date1), year(date1)+offset) ;

/* diff1 = 7, the number of days between
12/20/1996 and 12/27/1996. */
diff1 = intck('day', date1, date3) ;

/* diff2 = 0, the number of months between
12/20/1996 and 12/27/1996. */
diff2 = intck('month', date1, date3) ;

/* diff3 = 2, the number of years between
12/20/1994 and 12/20/1996. */
diff3 = intck('year', date7, date1) ;

/* diff4 = -2, the number of years between
12/20/1996 and 12/20/1994. */
diff4 = intck('year', date1, date7) ;

/* diff5 = 1, the number of years between
1/19/1995 and 12/20/1996. */
diff5=intck('year',date7+30,date1) ;
```

5.3. Sorting SAS date values

Since SAS date values are numeric values that represent the number of days before or after January 1, 1960, you can sort them with PROC SORT just like any numeric variable.

5.4. Writing SAS date values

5.4.1. Formats: the basics

A format is an instruction that tells the SAS system how to write data values. The SAS system provides many formats to write SAS date values.

The appropriate format depends upon how the SAS date values should be written. For example, if a variable is set to the SAS date value for December 20, 1996, different formats are needed to write it each of the following ways.

```
20DEC96
20DEC1996
12/20/96
12/20/1996
20 December 1996
96-12-20
19961220
```

SAS date values are usually written using a format, because formatted values (e.g., 12/20/1996 or 20DEC1996) look more meaningful than actual SAS date values (e.g., 13503 for December 20, 1996).

Formats are documented in the *SAS Language Reference, Version 8*. A summary table by category on pages 66-70 provides a useful quick reference.

Formats can be associated with a variable temporarily or permanently.

1. A *temporary* format is associated with a variable for a single PROC or DATA step. This is demonstrated below for a DATA step and PROC PRINT.

2. A *permanent* format is associated with a variable until you explicitly change or remove the format. Subsequent DATA and PROC steps use the format to print the variable.

Assign a permanent format with PROC DATASETS or a DATA step. If you do not otherwise need to read the data set, PROC DATASETS is more efficient because it updates the format without reading each observation.

Example. Assign a permanent format, YYMMDDN8., to the variable dat1 in the WORK data set ONE.

With PROC DATASETS

```
proc datasets library=work ;
   modify one ;
   format dat1 yymmddn8. ;
run;
```

In a DATA step.

```
data one;
   DATA step statements
   format dat1 yymmddn8.;
   more DATA step statements
```

```
run;
```

5.4.2. Writing SAS date values in a DATA step

As noted above, SAS date values are usually written using a format. In a DATA step, two ways to write SAS date values using a format are as follows.

1. Permanently associate a format with a variable that contains SAS date values, using PROC DATASETS or a FORMAT statement. PUT statements automatically use the format to print the variable.

Example. dat1 is printed using the YYMMDDN8. format in this and subsequent DATA and PROC steps.

```
data one;
    /* If a permanent format is already associated with
    dat1 (via PROC DATASETS or a prior DATA
    step), omit the FORMAT statement here. */
    format dat1 yymmddn8.;
    put dat1;
    more DATA step statements
run;
```

2. Use a temporary format in a PUT statement. The format is not permanently associated with the variable. A temporary format in a PUT statement overrides a format permanently associated with a variable, if one exists.

The following table shows the text written to the SAS log from a DATA step when the variable dat1, which contains the SAS date value for December 20, 1996, is printed using different formats.

SAS statement	Text written to SAS log
dat1=mdy(12, 20, 1996) ;	(nothing written)
put dat1 ;	13503
put dat1 date7. ;	20DEC96
put dat1 date9. ;	20DEC1996
put dat1 mmddyy8. ;	12/20/96
put dat1 mmddyy10. ;	12/20/1996
put dat1 worddatx. ;	20 December 1996
put dat1 yymmdd8. ;	96-12-20
datetemp =compress (put (dat1,yymmdd10.),-') ; put datetemp $8. ;	19961220
put dat1 yymmddn8.;	19961220

Notes on the table displayed above.

1. For the first PUT statement, 13503 is the SAS date value for December 20, 1996.

2. The next to last example illustrates a common workaround to a Version 6 limitation: no direct way to write SAS date values of the form yyyymmdd (e.g., 19961220). The YYMMDD8. format writes yy-mm-dd output (96-12-20) and the YYMMDD10. format writes yyyy-mm-dd output (1996-12-20). The first statement generates 1996-12-20, compresses out the dashes, and sets the variable datetemp to the resulting character string, 19961220. The second statement writes 19961220.

3. The last example illustrates a format available in Version 8 that directly writes SAS date values in the yyyymmdd format. To simplify older code akin to the next to last example, switch to the new format.

5.4.3. Writing SAS date values with PROC PRINT

PROC PRINT prints all or part of a SAS data set. As noted above, SAS date values are usually written using a format. Two ways to specify that a PROC PRINT step writes SAS date values using a format are as follows.

1. In PROC PRINT, use a FORMAT statement to temporarily (for that PROC PRINT only) associate a format with a variable that contains SAS date values.

2. Use PROC DATASETS or a FORMAT statement to permanently associate a format with a variable that contains SAS date values. See section 5.4.1, “Formats: the basics,” for an example. Subsequent DATA and PROC steps, including PROC PRINT, utilize the format to print the variable.

The following example illustrates the use of formats in PROC PRINT. sasdat1, sasdat2, and sasdat3 are SAS date values. In the DATA step, the YYMMDDN8. format is permanently associated with sasdat1. In the first PROC PRINT, sasdat1 is printed using a permanent format and sasdat2 is printed using a temporary format. In the second PROC PRINT, only sasdat1 is printed with a format.

```
data dates1;
    input @1 sasdat1 yymmdd8.;
    sasdat2 = sasdat1;
    sasdat3 = sasdat1;
    /* associate a permanent format with sasdat1 */
    format sasdat1 yymmddn8.;
    datalines;
19991231
19870219
```

```
; run;

proc print data=dates1 ;
     /* associate a format with sasdat2, this step only */
     format sasdat2 date9.;
run;

proc print data=dates1 ;
/* no format is associated with sasdat2 in this step */
run;
```

Results for first PROC PRINT:

OBS	SASDAT1	SASDAT2	SASDAT3
1	19991231	31DEC1999	14609
2	19870219	19FEB1987	9911

Results for second PROC PRINT:

OBS	SASDAT1	SASDAT2	SASDAT3
1	19991231	14609	14609
2	19870219	9911	9911

5.4.4. Formats and informats: details

Replace a format or informat with another format or informat

Section 5.4.1, "Formats: the basics," showed how to assign a permanent format with PROC DATASETS or a DATA step. Use the same methods to assign a permanent informat. If the variable already has a format or informat, it is overwritten.

Remove a format or informat

Remove a format or informat with PROC DATATSETS or a DATA step. Use a FORMAT or INFORMAT statement but do not specify a format or informat.

Example. Two ways to remove the format and informat for the variable sasdat1 in the WORK data set ONE are as follows.

With PROC DATASETS.

```
proc datasets library=work ;
     modify one ;
     format sasdat1 ;
     informat sasdat1 ;
run;
```

In a DATA step.

```
data one;
     set one;
     format sasdat1 ;
     informat sasdat1 ;
run;
```

Print a format or informat

Three ways to print the format or informat for the variable sasdat1 in the WORK data set ONE are as follows.

1. Use PROC CONTENTS to print information about all variables in the data set, including formats and informats.

```
proc contents data=one;
run;
```

2. Use PROC PRINT to print format and informat information from the data dictionary tables.

```
proc print data=sashelp.vcolumn;
     where libname="WORK" and memname="ONE"
       and name="sasdat1";
     var format informat;
run;
```

3. Use PROC SQL to read the data dictionary tables and create a data set. Use PROC PRINT to print the data set.

```
proc sql;
     create table new as
     select format, informat
     from dictionary.columns
     where libname="WORK" and memname="ONE"
       and name="sasdat1";
proc print data=new;
run;
```

6. SAS date values and the year 2000

This section describes a few problems with SAS date values related to the year 2000, and how to fix them.

6.1. Convert yymmdd data to SAS date values if yy is 00

To convert numeric variables of the form yymmdd to SAS date values, use the Z6. format instead of the 6. format to prevent errors when yy is 00, as in the years 1900 and 2000.

Example. The following DATA step converts numeric values with a yymmdd format to SAS date values.

```
data two;
     set one;
     sasdate1 = input (put (date1,6.), yymmdd6. ) ;
     more DATA step statements
run;
```

The following values of date1 are handled properly.

```
980324
991031
10101 (January 1, 2001)
```

The YYMMDD6. informat cannot accept values with fewer than 5 characters, so for values of date1 in the year 2000, such as the following, sasdate1 is a missing value.

```
101   (January 1, 2000, note that numeric values do
      not include leading zeros)
205   (February 5, 2000)
1231  (December 31, 2000)
```

To fix the problem, use the Z6. format instead of the 6. format, because Z6. generates leading zeros. For example, for 101 (January 1, 2000),

```
put (date1,6.) generates "101"
put (date1,z6.) generates "000101"
```

This statement generates correct results in all cases.

```
sasdate1 = input (put (date1,z6.), yymmdd6. ) ;
```

6.2. Create Julian dates with the JULDATE function for dates beginning 1/1/2000

The JULDATE function creates a Julian value from a SAS date value, as follows.

> For dates within the 100 year range defined by YEARCUTOFF, which by default is January 1, 1920 - December 31, 2019, JULDATE creates a 5-digit result, such as 99001 for January 1, 1999.
>
> Otherwise, JULDATE creates a 7-digit result, such as 2099001 for January 1, 2099.

Values beginning January 1, 2000 may return undesired results. For example, in the following code, DATE1 is the SAS date value for January 1, 2000. julian1 gets the value 1, because the JULDATE function returns the numeric value 00001, and the leading zeros are truncated.

```
date1="01jan2000"d;
julian1 = juldate(date1);
```

To fix the problem, use the JULDATE7 function instead of the JULDATE function. JULDATE7, which is new in Version 8, returns a seven-digit Julian date in all cases. In the following code, julian1 gets the value 2000001.

```
date1="01jan2000"d;
julian1 = juldate7(date1);
```

Another solution is to use the JULIAN. informat, as in the following code, which sets julian1 to 2000001.

```
date1="01jan2000"d;
julian1 = input(put(date1,julian7.),8.);
```

7. Dates not represented as SAS date values and the year 2000

Instead of using SAS date values, calendar dates are sometimes stored as one of the following.

> A numeric value, e.g., 19990703, 199907, or 990703.
>
> A character value, e.g., "19990703", "199907", or “990703".
>
> Multiple numeric or character values containing parts of a date that represent a date if combined or concatenated.

When calendar dates are not stored as SAS date values, errors can occur, some of which are due to the year 2000. This section illustrates a few errors and how to fix them.

7.1. Example 1. Error calculating a date

In two years, gnp doubles. Two new variables are created: date2 by adding two years to the current date, and gnpnew by multiplying gnp by 2. In the second observation, date2 has an incorrect value.

```
data badone ;
     input     @1 date1
               @8 gnp ;
     datalines;
960101 100
990102 200
; run;

data badtwo ;
     set badone ;
     addyears = 2 ;
     date2 = addyears * 10000 + date1 ;
     gnpnew = gnp * 2 ;
     put "on " date1 "gnp was " gnp
       "but on " date2 " gnp is " gnpnew ;
run;
```

The following output is written to the SAS log.

```
on 960101 gnp was 100 but on 980101  gnp is 200
on 990102 gnp was 200 but on 1010102  gnp is 400
```

To fix the problem, use SAS date values, as follows. The YEARCUTOFF option determines the century of the dates.

```
data badone ;
    input     @1 date1 yymmdd6.
              @8 gnp ;
    datalines;
960101 100
990102 200
; run;

data badtwo ;
    set badone ;
    addyears = 2 ;
    if day(date1) = 29 and month(date1) =2 and
        mod(addyears,4) ne 0
    then date2 = mdy(2, 28, year(date1)+addyears) ;
    else date2 = mdy(month(date1),
        day(date1), year(date1)+addyears) ;
    gnpnew = gnp * 2 ;
    put "on " date1 yymmddn8. " gnp was " gnp
      "but on " date2 yymmddn8. " gnp is " gnpnew ;
run;
```

The following output is written to the SAS log.

```
on 19960101 gnp was 100 but on 19980101 gnp is 200
on 19990102 gnp was 200 but on 20010102 gnp is 400
```

Note that if you convert the dates to a variable of the form yyyymmdd instead of SAS data values, you must still make an assumption about the range of the dates (analogous to YEARCUTOFF for SAS date values) to determine the century. For example, the following statements convert a numeric variable of the form yymmdd to a numeric variable of the form yyyymmdd for different date ranges.

If the dates are assumed to be in the range 1900 - 1999:

```
datenew = dateold + 19000000 ;
```

If the dates are assumed to be in the range 1925 - 2024:

```
if dateold >= 250000 then
   datenew = dateold + 19000000 ;
else
   datenew = dateold + 20000000 ;
```

If the dates are assumed to be in the range 1950 - 2049:

```
if dateold >= 500000 then
   datenew = dateold + 19000000 ;
else
   datenew = dateold + 20000000 ;
```

7.2. Example 2. Error printing a date in a report

In each observation, two years are added to the year variable, yy, and the year is printed at the top of the page. In the third observation, yy has an unexpected value, 101.

```
data baddate ;
    input     @1 yy
              @4 gnp ;
    datalines;
96 100
59 200
99 300
;run;

data badthree ;
    set baddate ;
    file 'file-specification';
    addyears = 2 ;
    yy = yy + addyears ;
    put  _page_
         @20 "Year= " yy 2. ;
    more DATA step statements
run;
```

The following text is written at the top of the three pages of output, starting in column 20. "**" is written instead of the number 101, because 101 is too large for the 2. format.

```
page 1:   Year= 98
page 2:   Year= 61
page 3:   Year= **
```

When "**" is written to the file, the following message is written to the SAS log. Users might not notice the message, because it is a NOTE, not an ERROR or WARNING, and because it does not include the unprintable value or the observation number for which the problem occurred.

```
NOTE: At least one W.D format was too small for the
number to be printed. The decimal may be shifted by
the "BEST" format.
```

To fix this problem, use SAS date values for January 1 (since only the year is available). Replace yy = yy + addyears ; and the PUT statement in the second DATA step with the following code to print four-digit years.

```
date1 = mdy(1, 1, yy); /* make Jan 1 SAS date value */
date1 = intnx('year', date1, addyears) ; /* offset */
tempyear = year(date1);
put  _page_
     @20 "Year= " tempyear 4. ;
```

To print two-digit years, use the following code instead.

```
date1 = mdy(1, 1, yy); /* make Jan 1 SAS date value */
date1 = intnx('year', date1, addyears) ; /* offset */
tempyear = substr(put(year(date1), 4.), 3, 2);
put  _page_
     @20 "Year= " tempyear $2. ;
```

8. When part of the date is missing

In the previous example, only the year was available, so SAS date values for January 1 were created. Suppose one or more date components (day, month, year) could be missing in some observations (i.e., missing components, if any, could vary from observation to observation). One solution (suggested by Ian Whitlock and Kathy Fraeman) is to maintain two variables: a SAS date value and a status variable equal to either 0 (date information complete) or the sum of the following values as appropriate.

0	all date information available
1	day missing
2	month missing
4	year missing

For example, if the status variable is 2, only the month is missing for that observation. If the status variable is 3, the day and month are missing for that observation (and the SAS date value is set to January 1 of the year).

9. SAS time and datetime values

A SAS time value is a number whose value is the number of seconds since midnight, independent of a date.

A SAS datetime value is a number whose value is the number of seconds before or after January 1, 1960.

As with SAS date values, some formats, informats, and DATA step functions are specific to or useful with time and datetime values.

For more information, see the chapter *Dates, Times, and Intervals* in the *SAS Language Reference: Concepts* and the chapters on formats, informats, and functions in the *SAS Language Reference, Version 8*.

CONCLUSION

This paper discussed date handling in the SAS system. It focused on SAS date values; how to create, transform, and print them, and their interaction with the YEARCUTOFF option, formats, and informats. Hopefully, this paper made SAS date values easy to understand and use.

For more information, contact

Bruce Gilsen
Federal Reserve Board, Mail Stop 157
Washington, DC 20551
phone: 202-452-2494 e-mail: bruce.gilsen@frb.gov

REFERENCES

Elam, Amber H. (1996), "SAS Dates: Everything You Always Wanted to Know, Including the Year 2000," Observations: The Technical Journal for SAS Software Users,5(4),23-30.

Gilsen, Bruce (1999), "SAS Software and the Year 2000," internal manuscript, Federal Reserve Board.

Langston, Richard and Williams, Chris (1997), "The Year 2000: Preparing for the Inevitable," Proceedings of the Tenth Annual NorthEast SAS Users Group Conference.

SAS Institute Inc (1999), "SAS Language Reference: Concepts," Cary, NC: SAS Institute Inc.

SAS Institute Inc (1999), "SAS Language Reference, Version 8," Cary, NC: SAS Institute Inc.

SAS Institute Inc (1999), "SAS Macro Language: Reference, Version 8," Cary, NC: SAS Institute Inc.

SAS Institute Inc (1999), "SAS Procedures Guide, Version 8," Cary, NC: SAS Institute Inc.

Taubman, Steve (1996), "How to Prepare for the Year 2000," in the Proceedings of the Ninth Annual North East SAS Users Group Conference, 593 - 595.

Whitlock, H Ian (1999), “Managing the INTNX Function,” in the Proceedings of the 1999 SouthEast SAS Users Group Conference.

ACKNOWLEDGMENTS

The following people contributed extensively to the development of this paper: Kathy Fraeman, Donna Hill at the Federal Reserve Board, Avi Peled at the Federal Reserve Bank of Philadelphia, and Ian Whitlock at Westat. Their support is greatly appreciated.

TRADEMARK INFORMATION