WORKING WITH SAS® DATE AND TIME FUNCTIONS

Andrew H. Karp
Sierra Information Services, Inc.
Sonoma, California USA

Introduction

Most SAS Software users need to work with data sets that contain one or more variables (columns) containing the date and/or time at which an event occurred. This paper describes some of the core SAS System tools you can use with your date and time data, and discusses how these tools can simplify many of the tasks commonly performed on this type of variable. We will also discuss some of the new features for date and time variables that have been added in recent releases of SAS Software, as well as some enhancements you can expect in the forthcoming release of SAS 9 and SAS 9.1

SAS users often need to determine the

- frequency that a phenomenon of interest occurs in time (e.g., how many vehicles crossed the bridge in May 2003?)
- number of time intervals which have elapsed between two events (e.g., how many days elapsed between surgery and patient discharge?)

In addition, SAS Software users need to

- operate conditionally on the observations based on the value of date or time variables. (e.g., select only claim records where the submission date is prior to January 1, 2003)
- aggregate observations from a higher time frequency interval to a lower frequency interval (e.g., daily to monthly)
- interpolate missing values in a time series prior to aggregation or analyzing the series using tools available in, for example, SAS/ETS™ software.
- interpolate higher frequency interval observations from data collected at lower frequencies (e.g., estimate weekly values from a series of data collected monthly)
- create a SAS date, time, or datetime variables from "raw data" during Data Step.

The SAS System provides a wealth of tools for users who need to work with data collected in the time domain.

These tools can be broken down in to several broad categories:

- Informats
- Functions
- SAS Programming Language Elements
- SAS System Options
- Procedures
- Macro Symbol Table Values

We will explore the tools in each of these categories later in the paper. In order to to take advantage of them, we need to first understand the core concepts about how the SAS System stores the values of variables representing dates and times.

Core Concepts

A SAS **date, time or datetime variable** is a special case of the numeric variable. The values of a **date variable** represent the number of days before or after January 1, 1960, and the values of a **time variable** are the number of seconds since midnight. In the SAS System, a date value of zero (0) represents January 1, 1960 and a time value of zero (0) represents midnight.

A time variable is independent of a date variable. Similarly, a SAS **datetime** variable contains both date and time information (e.g., January 20, 1994 at 4:13 p.m.) as the number of seconds from midnight at January 1, 1960.

Many SAS users who work with data obtained by using SAS/ACCESS software to RDBMS products such as SYBASE or ORACLE will need to understand how to work with SAS datetime variables.

A SAS date value is a constant that represents a fixed value. Valid SAS dates are from 1582 A. D. (following adoption of the Gregorian Calendar) to 20,000 A. D.

Since SAS stores the values of dates and times a numbers, a format is almost always required when portraying them in SAS-generated output (see below).

Informats: *Creating SAS Date Variables from Raw Data*

Many raw data files contain columns (variables) with date values written in, say, MMDDYY or MM/DD/YYYY format. For example, the value in the field may be 01MAY2003 or 01/05/2003. In some situations the European method of writing dates, DDMMYY, might be used, so the value of the field might appear as 05/01/2003.

Broadly speaking, an Informat gives the SAS System instructions on how to convert fields in a raw data file to variables in a SAS data set. Informats are placed in the INPUT Statement and SAS uses them when making variables in SAS data sets from the fields/columns in your raw data file.

There are several Informats available that are designed to transform columns of raw data containing date fields in to the values of SAS date, time or datetime variables. To use them correctly, you MUST know how the date field is structured in the raw data file. Otherwise, you are likely to make a major, and possibly undetectable, error in your program. Specifying the MMDDYY10. Informat instead of the DDMMYY10. Informat, for example, means that the first two values of the raw data field represent the month, rather than the day of the month. Take the time to understand your raw data values before applying SAS Informats in your INPUT Statements!

Example: Using the YYMMDD8. Informat
A variable in a raw data set which has the format YYMMDD8. (e.g., 20030425 for April 15, 2003) can be converted to a SAS date variable by using the YYMMDD8. ***informat*** in the Data Step. Using this ***informat*** 'converts' the values of the raw data field to SAS date variable. Other commonly used SAS date ***informats*** include DDMMYY8. , MMDDYY8. and DATETIME18.

Example: ANZAC Day (Australia) 2003
If a value of a variable raw data set to be read by the SAS System contained a value 2003/04/25, the YYMMDD10. Informat can be used to create a SAS date variable by placing the informat immediately adjacent to the variable name in the INPUT statement.

INPUT ANZACDAY `YYMMDD10.`;

The SAS System will automatically convert the text representation of the raw data variable in to a SAS date variable with a value of 15820, the number of days between 1 January 1960 and 25 April 2003.

The ANYDATE Informats (SAS 9.1)
Even though SAS provides a wide range of Informats for many different raw data structures representing dates and times, there are still some situations where the structure of the raw data is not accommodated by the SAS-supplied Informats.

The forthcoming release of SAS 9.1 will include the new ANYDATE Informats, which will make it even easier to convert your raw data fields in to SAS date, time or datetime variables. Use of the ANYDATE Informats will be aided by the new (to SAS 9) DATESTYLE SAS System Option, which is discussed below.

Functions for Date, Time and Datetime Variables

The SAS Programming Language includes several functions that are useful when working with your date, time and datetime variables. These functions can be used to:

- Create SAS date, time or datetime variables
- Extract the 'parts' from an existing date, time or datetime variable
- Determine the number of time intervals which have elapsed between two events
- Assign the value of a variable to some future date.

Functions That Create SAS Date, Time or Datetime Variables

- **MDY Function:** this function creates a SAS date variable from three arguments giving the month, day and year. This function is particularly useful if you have three separate variables in your SAS

data set that contain the month, day and year, and you want to create a SAS data set from these values.

- **Example**: **`Date_var=MDY(Monthvar, Dayvar, Yearvar);`**
- The MDY Function is often used to create a SAS date variable with the first (or last) day of the month in which the event of interest occurred. For example, you may have a SAS date variable representing the date on which a transaction occurred, and you want to "roll up," or aggregate all of the transactions by month. Since you can "nest" multiple SAS programming language functions in a single SAS assignment statement, you could easily create a new variable representing the first day of the month using the following syntax:
- **`Trans_month = MDY(MONTH (Trans_Date),1, YEAR(Trans_Date));`** *[The MONTH and YEAR Functions are discussed below.]*

- **YYQ Function:** A SAS date variable with the value of the first day of the calendar quarter is created by this function.
  - **Example: `Quarter = YYQ(Yearvar,Qtr_var);`**

- **DHMS Function:** This function creates a SAS datetime variable from arguments giving the SAS date value, hour, minute and seconds. If you wanted to create a SAS datetime variable with the value of noon on December 25, 2002, the syntax would be:
  - **`XMAS_NOON = DHMS (MDY (12,25,2002),12,00,00);`**
  - The value returned by this assignment statement is 1356436800, which is the number of seconds from midnight January 1, 1960 to noon on December 25, 2002.

A constant can be substituted for any of the variables in the function. In the "first day of the month" example above, the constant '1' was supplied as the second argument to the MDY function.

Functions that Extract 'Parts' from a SAS Date, Time or Datetime Variable

Several SAS programming language functions are available to extract a "part" or "piece" of a SAS date, time or datetime variable. These include:

- For SAS date variables:

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

- ***Example: July 4, 2003***

Assume we have a data set with SAS date values for all United States federal holidays in 2003. The dates are stored in a variable called HOLIDAY_DATE. For July 4, 2003, the SAS date value is 15890. Here is what you will obtain by applying various SAS date functions to that value:

Syntax	*Value Returned by Function*
`Month(Holiday_Date)`	**7**
`Day(Holiday_Date)`	**4**
`Year(Holiday_Date)`	**2003**
`QTR(Holiday_Date)`	**3**
`Weekday(Holiday_Date)`	**6**

- For SAS Time Variables

HOUR	Returns the hour
MINUTE	Returns the minute
SECOND	Returns the second

- ***Example: 9:17 and 43 seconds PM***

Assume that we have a variable giving us the time at which an event occurred. For the example time, the SAS time variable value would be 76663, which is the number of seconds which elapsed between midnight (time variable value zero) and the given time. Here is

what you will obtain by applying various SAS time functions to that value: (Notice that hour values are given using "military time.")

Syntax	*Value Returned by Function*
`Hour(Time_Var)`	**21**
`Minute(Time_var)`	**17**
`Seconds(Time_var)`	**43**

- For SAS Datetime variables

The date and or time elements (or 'parts') of a SAS datetime variable can be extracted using the DATEPART and TIMEPART functions. Suppose you are working with data from a hospital admissions data set, where each admission record is "timestamped" with the date and time the patient's record is entered in to the system. For this example we will assume that the date and time of admission is stored as a SAS date variable called ADMISSION. So, for an admission that occurred at 9:17 and 43 second PM on May 3, 2003, the timestamp value is 1367615683, the number of seconds between midnight on January 1, 1960 and the given date. It's obviously impossible to figure out the "date" or the "time" part of this variable's value without some help.

Here is what you will obtain by using the DATEPART and TIMEPART functions to the datetime variable value give above:

Syntax	*Value Returned by Function*
`Datepart(Admission)`	**15828**
`Timepart(Admission)`	**76663**

Using SAS Formats for date, time or datetime variables will portray their values in a more comprehensible way. This topic is discussed in detail in a subsequent part of this paper.

Declaring Date, Time or Datetime Constants

You can easily declare a date, time or datetime value using SAS Programming Language elements specifically designed for that purpose. These language elements eliminate the need for you to calculate the SAS date, time or datetime value that you want to specify.

- Declaring a date constant. Suppose you want to apply a "Subsetting IF" statement in a Data Step so that only observations with a value of the variable CHARGE_DATE that occurred *after* May 1, 2002 will remain in the Program Data Vector (PDV) for additional processing. All you need to write is:

```
IF Charge_Date > '01May2003'D;
```

The letter "D" next to the text string with the day, month and year instructs the SAS System to convert the string to the appropriate SAS date value. So, when the data step processes your data, what the SAS System does is compare the value of the numeric variable Charge_Date to the numeric value of the SAS date constant. Notice that the text string is in day/month/year format.

- A time constant can be specified as a text stirng, too. Here is an example:

```
If Charge_Time > '14:35:00'T;
```

As with the date constant example given above, the letter "T" next to the text string with the hour, minutes and seconds instructs the SAS System to calculate the appropriate SAS time value (number of seconds from midnight) represented by the text string.

- Specifying a datetime constant is also possible. For example:
-

```
If Admit_DateTime >
'01May2002:12:30:00'DT;
```

By now you've probably figured out that SAS will calculate the appropriate datetime value (number of seconds from midnight on January 1, 1960).

Functions for Determining Duration vs. Direct Calculation of Duration

A common application of SAS System date and time processing capabilities is to determine how long a period has elapsed between two points in time. This can be accomplished by one of two methods:

- arithmetic operation (usually subtraction and/or division) between two SAS date, time or datetime variables or between a SAS date, time, or datetime variable and a constant term
- use of the **INTCK** ('in-tick') function

Arithmetic Operation

The number of days which have elapsed between two points in time is easily determined by subtracting the value of one SAS date variable from another, or by subtracting a SAS date variable from a SAS date constant (or vice versa, as may be appropriate). The result can then be divided by an appropriate constant to obtain the desired number of time periods between the two values. A common requirement is to determine how many years have elapsed between two time periods:

YEARS = (date2 - date1)/365.25;

returns the number of units of 365.25 which have occurred between the two date variables. This is a commonly accepted practice to determine the number of years occurring between two points in time. Similarly, 30.4 is frequently used as the denominator to convert the number of days to the number of months.

INTCK Function

A popular and powerful SAS function, **INTCK**, is available to determine the number of *time periods* which have been *crossed* between two SAS date, time or datetime variables. The form of this function is:

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

Where:

'interval' = character constant or variable name representing the time period of interest enclosed in single quotes

from = SAS date, time or datetime value identifying the start of a time span

to = SAS date, time or datetime value identifying the end of a time span

This function will return the number of time periods which have occurred (i.e., have been *crossed*) between the values of the *from* and the *to* variables.

Time Intervals

Among the time intervals you can specify as the first argument to the INTCK Function are: MONTH, DAY, YEAR, DECADE, HOUR, MINUTE and SECOND.

In addition, four new date and datetime intervals were implemented in Release 6.07 of SAS System Software. They are:

WEEKDAY: counts the number of weekdays between two time values, with the weekend days counted as part of the preceding weekday. By default, Saturday and Sunday are considered "weekends."

TENDAY: counts the number of ten-day intervals between two time values. The default is for the month to be broken in to three periods: a) first through tenth days, b) eleventh through twentieth day, c) twenty-first day through end of the month.

SEMIMONTH: breaks each month in to two periods, starting on the first and sixteenth days of the month.

SEMIYEAR: specifies semiannual periods of six months.

Arithmetic Operation vs. INTCK Function

Important differences exist between how these two methods determine the number of periods of time that have elapsed between events. These differences can be demonstrated as follows. Suppose a child is born (and therefore 'admitted' to the hospital) on December 28, 2002 and discharged on January 2, 2003. The child is therefore five days old at discharge. Subtracting SAS date value for discharge date from the SAS date value for admission date
yields 5, which is the desired result. But, how many *years* old is the child? An acceptable estimated answer is 5/365.25, or .02 *years.* But, using the INTCK function, with the YEAR interval argument

```
AGE=INTCK('YEAR', Admit_date,
Discharge_Date);
```

returns 1 as the result. Why? Because the INTCK function counts the number of time *intervals* which have been *crossed* between the *from* and *to* expressions arguments of the function. Since YEAR was specified as the desired *interval*, and January 1 is 'enclosed' between the *from* and *to* arguments, the child's

age is given by the result of the INTCK function to be 1"year", rather than 5 "days".

Users should take in to account the important differences in results which will occur from using one or the other of these approaches and make sure that the one the apply is appropriate for their particular data processing/analysis.

The INTNX Function

Also useful is the INTNX function, which *creates* a SAS date, time or datetime value that is a given number of time intervals from a starting value. The form of this function is:

INTNX('interval',from,to)

Where *interval, from* and *to* have the same meanings and definitions as for the INTCK function described earlier. For example, suppose a hospital wanted to send a postcard to the parents of newborns three months after the child is born reminding them to schedule a follow-up visit. Using:

MAILDATE = INTNX('month',BDATE,3);

Where BDATE is a date variable representing date of birth, the INTNX function will return values of MAILDATE which are the SAS date values for the first day of the month which is three months past the child's birthday. Thus, the INTNX and INTCK functions operate in a similar fashion by counting the date *boundaries* which are *crossed.*

Alternatively, if the reminder postcard is to be generated *90 days* after the child is born, a statement such as:

MAILDATE = BDATE + 90;

will assign the value of the variable MAILDATE equal to the child's birthday plus 90 days, or the number of days from January 1, 1960 to the child's birthday plus an additional 90 days. This result differs from that obtained by using the INTNX function, which by default would return the SAS date value for the *first day of the month* three months after the child was born.

Enhancements to the INTNX Function in Release 6.11 of the SAS System.

Prior to Release 6.11 the INTNX function returned a value representing the *beginning* of the interval specified in the function's third argument. For example, the value returned by specifying MONTH as the desired interval was the first day of the month.

Starting with Release 6.11, an optional fourth argument is available for the INTNX function. Users can specify BEGINNING (the default), MIDDLE, or END. Using the MIDDLE argument returns a value representing the period's midpoint and END returns a value representing the end of the period. For example, if a user wanted to advance a date value from January 1, 2003 to June 30, 2003, the END argument would be applied as follows:

```
NEWVAR =
INTNX('Month','01Jan2003'd,5,'END')
;
```

The END alignment argument is quite useful when creating a SAS date variable representing the last day of the month in which

Other Useful Functions

The table below describes other SAS Programming Language Functions that may be useful when working with date, time, or datetime variable processing.

Function Name	Description
`Today()`	Returns the SAS date value from the system clock
`Time()`	Returns the SAS time value from the system clock
`Datetime()`	Returns the SAS datetime value from the system clock

Using these functions can avoid the need to "hard code" date values in your programs. For example, if you need to run a production program that, say, calculates the number of days between today's date and the date an insurance claim was entered in to the data base, you could write the following assignment statement in a Data Step:

```
Days_Elapsed = TODAY() -
Enter_Date;
```

SAS will read the SAS date value from the system clock when executing your Data Step.

Compound Statements

Combining SAS date, time and datetime statements in to a single expression can reduce processing time and programming steps. Here are two examples:

Compound Subsetting IF Statement:
Example 1

Two or more conditions for a SAS date, time or datetime variable can be tested in the same "Subsetting IF" or assignment statement. For example, if a researcher working on a hospital admissions data set desired to only analyze records where patients were admitted in the third quarter of 2001, she could write:

```
IF YEAR(ADMIT) = 2001 and
      QTR(ADMIT) = 3;
```

Compound Subsetting IF Statement:
Example 2

As with the previous example, placing all subsetting conditions in a single programming statement often enhances program performance. If the variable ADMIT was a datetime variable, the following compound statement would select only those observations where the patient was admitted on a Saturday or Sunday after 4:00 pm during the second quarter of 2002:

```
IF YEAR(DATEPART(ADMIT)) = 2002
        AND
QTR(DATEPART(ADMIT)) = 2
        AND
WEEKDAY(DATEPART(ADMIT)) IN(1,7)
         AND
HOUR(TIMEPART(ADMIT)) > 16;
```

Notice that the above statement operates from 'largest' to 'smallest' time interval: year, quarter, weekday, hour. This arrangement helps reduce the average amount of time each observation from the "source" data set will spend in the Program Data Vector, and depending on the number of observations, in the data set, may reduce program processing time. .

Formats: Altering External Representations of SAS Date, Time and Datetime Variables

Treating date, time and datetime variables as numeric variables makes it easier for the SAS System to operate on them than if they were character variables. This, however, makes in nearly impossible for a (human) end-user to discern the values of these variables and to represent them in a meaningful way in reports or other output.

This problem is easily solved by appropriate use of one of dozens SAS **formats** for date, time or datetime variables. Among the commonly used formats are:

Format:	**Result:**
MMDDYY10.	07/04/1997
DDMMYY10.	04/07/1997
WORDDATE18.	July 4, 1997
WEEKDATE29.	Friday, July 4, 1997
MONYY5.	JUL97
MONYY7.	JUL1997

Two new date formats were added in Release 8.2 of the SAS System that are designed to ease the portrayal of datetime variable values. These formats might, depending on your specific programming/analytical requirements, eliminate the need to use the DATEPART programming language function, described above, to obtain the date "part" of a SAS date time variable as a separate variable in your data set.

For example, here is how these two new formats would portray the SAS datetime value of 1367411430, or May 1, 2003 at 12:30 pm. The table below also shows the representation of this value using the Datetime20. format for comparison.

Format:	**Result:**
Datetime20.	**01MAY2003:12:30:30**
DTDATE.	**01MAY03**
DTWKDATX.	**Thursday, 1 May 2003**

SAS Procedures: Aggregating and Interpolating Data

Data collected in the time domain can be **aggregated** from a lower to a higher period (e.g., monthly to yearly) using Base SAS software procedures such as FREQ, MEANS,

SUMMARY, or UNIVARIATE. The appropriate date, time or datetime variable(s) are used in either the CLASS or BY statements (or both, if appropriate).

Some analyses require that a time series be **interpolated** by estimating from higher-period observations to lower-period observations. Estimates of monthly values of a time series containing quarterly observations, may, for example, be desired.

Another key issue in aggregating and/or interpolating data collected in the time domain is treatment of missing observations. Some statistical techniques (e.g., Auto-Regressive Integrated Moving Average, or ARIMA models) for analyzing time series data are hampered by the presence of missing values; in other situations an analyst may want to estimate values for missing observations before performing aggregation and/or interpolation operations. In any event, substantial user intervention may be required in the Data Step to substitute 'appropriate' values for missing observations.

PROC EXPAND, in the SAS/ETS™ module, performs both aggregation **and** interpolation of data collected in the time domain, as well as estimation of the values of missing observations in the data set. This procedure can also apply a series of transformation operators to your data, either prior to, or after it performs the desired interpolation and/or aggregation. Among the transformation operations available in this procedure are forward and backward moving averages, moving sums, and other "moving time window" statistics.

If you have SAS/ETS software, take a few minutes to learn what PROC EXPAND can do for your data and time variable processing. You'll probably learn that using it can save you lots of time you'd otherwise spend writing long Data Steps to handle complex inter-observation processing of the observations in your data sets.

SAS System Options for Dates and Times

There are several Options that address how the SAS System works with dates and times.

- **NODATE**: This Option suppresses the printing of the date and time of SAS System initialization at the top of each page of output in your Output Window.
- **DTRESET**: New to SAS 9.1, this Option will print the current date and time in the Output window, rather than the date and time of SAS System initialization. This option will probably be helpful for users submitting very long jobs who need to know the precise time at which the job generated a specific piece of output.
- **YEARCUTOFF**: This option is used to assign the century to dates where two-digit years are used. Starting in Version 8, the default value of this option is 1920 (and remains the default in SAS 9, too).

The easiest (and least ambiguous) way to ensure your SAS date or datetime values refer to the appropriate century is to use a four-digit year value, rather than a two-digit value. For example, if your YEARCUTOFF system option value is set to 1920 (the default), the value of the variable **DATE='01JAN00'D** represents the number of days from January 1, 1900 to January 1, 2000, which is a negative number. If, however, the user set **DATE='01JAN1900'D**, SAS would set the value of the variable DATE to the number of days from January 1, 1900 to January 1, 1960. Taking the time to code those two extra keystrokes can potentially save you lots of aggravation.

Conclusion

There are many tools available in the SAS System to work with your date and time variables. From functions to formats, and from procedures to informats, users of SAS software will find it easy to work with date and time variables once they have mastered the core concepts presented in this paper.

References

Broder, John M., and Laurence Zuckerman, "Computers Are the Future But Remain Unready for It," *The New York Times,* April 8, 1997

Dilorio, Frank C., *SAS® Applications Programming: A Gentle Introduction,* Belmont, California: Duxbury Press, 1991

Langston, Richard D., and Chris Williams, *"The Year 2000: Preparing for the Inevitable,"* Proceedings of the Twenty-Second Annual SAS Users Group International Conference, Cary, NC: SAS Institute, Inc., 1997

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/ETS® User's Guide, Version 6, Second Edition,* Cary, NC: SAS Institute, Inc., 1993

SAS Institute, Inc., SAS® Technical Report P-222, *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 6.10,* Cary, NC: SAS Institute, Inc., 1994

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

The author may be contacted at:

Sierra Information Services, Inc.
19229 Sonoma Highway PMB 264
Sonoma, CA 95476 USA
707 996 7380
SierraInfo @ AOL.COM
www.SierraInformation.com