SESUG Paper 202-2018

Dating for SAS® Programmers

Joshua M. Horstman, Nested Loop Consulting

ABSTRACT

Every SAS programmer needs to know how to get a date... no, not that kind of date. This paper will cover the fundamentals of working with SAS date values, time values, and date/time values. Topics will include constructing date and time values from their individual pieces, extracting their constituent elements, and converting between various types of dates. We'll also explore the extensive library of built-in SAS functions, formats, and informats for working with dates and times using in-depth examples. Finally, you'll learn how to answer that age-old question... when is Easter next year?

INTRODUCTION

Our very existence as human beings is defined by the passage of time. It is no surprise, then, that we collect a lot of data about time. Fortunately, the SAS system is well-equipped to deal with information about dates and times. This paper will examine how SAS stores dates and times and will provide an overview of the many tools available in the SAS programming language for working with dates and times. These tools include formats, informats, functions, and automatic macro variables.

SAS DATE AND TIME BASICS

Variables in SAS data sets can only have one of two types: character or numeric. Unlike some other programming languages, there is not a separate variable type in SAS to hold dates and/or times. Instead, SAS handles dates and times by simply storing them as ordinary numbers. We will discuss three types of date and time values: dates, times, and datetimes.

DATE VALUES

A SAS date value is stored as the number of days since January 1, 1960. Thus, the date January 1, 1960, corresponds to a value of zero. Likewise, January 2, 1960, would be represented as 1.

Dates prior to January 1, 1960, are assigned negative numbers. Accordingly, December 31, 1959, is stored internally as -1. At present, the SAS system supports dates as far back as January 1, 1582, which translates to a SAS date value of -138,061. It can also handle future dates almost up to the year 20,000, which should suffice for most practical purposes.

TIME VALUES

Similarly, SAS stores time values as the number of seconds after midnight. Midnight itself is represented by a value of zero. One minute past midnight, which we call 12:01am, would be stored by SAS as 60. There is no need for negative numbers here since times prior to midnight are treated as part of the previous day. Thus, 11:59pm has a numeric value of 86,340.

DATETIME VALUES

Whereas a time value represents only a specific time of day without regard to the date, a datetime value corresponds with a specific time on a specific date. Datetime values are stored as the number of seconds since midnight at the beginning of January 1, 1960.

Thus, 12:01am on January 1, 1960, would translate to a datetime value of 60. The datetime value for 5:20pm on February 4, 1960, would be 3,000,000, since three million seconds would have elapsed since midnight on January 1, 1960. As of the writing of this paper, current SAS datetime values are approaching two billion, and will finally cross that threshold sometime in May 2023.

As with date values, datetime values are negative for timepoints prior to January 1, 1960. The datetime value for 11:59pm on December 31, 1959, is -60. The range of supported SAS datetime values is the

same as for date values: January 1, 1582 (datetime value of almost negative 12 billion) through roughly the year 20,000.

DATE, TIME, AND DATETIME LITERALS

Sometimes we wish to refer to a specific date, time, or datetime value as a constant within SAS code. This can be accomplished using a date, time, or datetime literal. Literals can be used in assignment statements, as function arguments, or anywhere one might use a variable containing a date, time, or datetime value.

A date literal is simply a quoted string (using either single or double quotes) containing the desired date with the letter "d" placed immediately after the quotes. The "d" tells SAS that this is a date literal as opposed to an ordinary character string. The date needs to be formatted in a specific manner using a one- or two-digit day, three-letter abbreviation for the month, and two- or four-digit year as shown below in Table 1 (SAS 9.4 Language Reference, pp. 99-100).

Time literals are indicated by a quoted string immediately followed by the letter "t". The quoted string must include the hours, minutes, and optionally seconds, separated by colons. The seconds may include fractional seconds using a decimal point. The hours may be specified based on a 24-hour clock. Alternatively, hours can be based on a 12-hour clock by appending a suffix of "am" or "pm" within the quotes.

Perhaps not surprisingly, datetime literals are specified by adding the letters "dt" after a quoted string. The required format is achieved by using the formats specified above for date and time literals concatenated together and separated by a colon.

Type of Literal	Required Format	Example
Date	`"ddmmm<yy>yy"D`	`"01JAN2018"D`
Time	`"hh:mm<:ss.s>"T`	`"13:30:00"T or "1:30:00pm"T`
Datetime	`"ddmmm<yy>yy:hh:mm<:ss.s>"DT`	`"01JAN2018:13:30:00"DT`

Table 1. Date, Time, and Datetime Literals

THE YEARCUTOFF OPTION

In the previous section, we saw that we can specify a date or datetime literal using only a two-digit year if desired. In order to store the date or datetime as a numerical value, SAS must decide which century was intended. This decision process is controlled by the YEARCUTOFF system option (SAS 9.4 System Options pp.335-336).

The YEARCUTOFF system option allows the user to indicate the first year of a 100-year span that will be used when assigning a century in cases where only a two-digit year is specified. The default value (as of SAS 9.4) is 1926. This means that two-digit years will be assumed to belong to the span from 1926 to 2025. In other words, the two-digit year "25" will be treated as 2025, while the two-digit year "26" will be considered as 1926.

As with all system options, the value of this option can be modified using the OPTIONS statement. The YEARCUTOFF option does not pertain only to the use of date and datetime literals but applies anytime a two-digit year is encountered when reading dates.

FORMATTING DATES AND TIMES WITH SAS FORMATS

The problem with storing dates and times as the number of days since January 1, 1960, and the number of seconds since midnight is that this isn't the way humans normally keep track of things. People don't say "I was born on SAS date 6,029," or "let's meet for lunch at 43,200".

Fortunately, SAS provides an extensive library of built-in formats that can be used to render numeric date, time, and datetime values as human-readable calendar dates and clock times. In addition, SAS also includes mechanisms for building custom formats that go beyond those which are built-in.

BUILT-IN FORMATS

A format provides a mechanism to display a data value using a specific pattern or set of instructions (SAS 9.4 Formats and Informats, p. 4). There are dozens of built-in formats in SAS 9.4 pertaining specifically to dates, times, and datetimes. A complete list of all built-in formats can be found in *SAS 9.4 Formats and Informats: Reference*.

These formats can be used wherever formats are supported in the SAS language. Methods for applying formats include the PUT statement, the PUT family of functions, the FORMAT and ATTRIB statements (which are available in both the DATA step and many procedures), and the %SYSFUNC macro function.

Built-In Date Formats

Table 2 lists some of the more commonly-used built-in date formats. The descriptions are adapted from *SAS 9.4 Formats and Informats: Reference*, and the output shows how the date value corresponding with December 31, 2018 (SAS date 21549) would appear when the format is applied.

Format	Description	Output
date5.	Writes date values in the form ddmmm.	**31DEC**
date7.	Writes date values in the form ddmmmyy.	**31DEC18**
date9.	Writes date values in the form ddmmmyyyy.	**31DEC2018**
date11.	Writes date values in the form dd-mmm-yyyy.	**31-DEC-2018**
day2.	Writes date values as the day of the month.	**31**
ddmmyy8.	Writes date values in the form dd/mm/yy.	**31/12/18**
ddmmyyd10.	Writes date values in the form dd-mm-yyyy.	**31-12-2018**
downame.	Writes date values as the name of the weekday.	**Monday**
e8601da.	Writes date values using ISO 8601 extended notation yyyy-mm-dd.	**2018-12-31**
mmddyy8.	Writes date values in the form mm/dd/yy.	**12/31/18**
mmddyyd10.	Writes date values in the form mm-dd-yyyy.	**12-31-2018**
monname.	Writes date values as the name of the month.	**December**
month2.	Writes date values as the number of the month.	**12**
monyy7.	Writes date values as the month and year in the form mmmyyyy..	**DEC2018**
qtr1.	Writes date values as the quarter of the year.	**4**
weekdate.	Writes date values as the day of the week and the date in the form day-of-week, month-name dd, yyyy.	**Monday, December 31, 2018**
weekdatx.	Writes date values as the day of the week and the date in the form day-of-week, dd month-name yyyy.	**Monday, 31 December 2018**
worddate.	Writes date values as the name of the month, day, and year in the form month-name dd, yyyy.	**December 31, 2018**
worddatx.	Writes date values as the name of the month, day, and year in the form dd month-name yyyy.	**31 December 2018**
year4.	Writes date values as the year.	**2018**
yymmddp10.	Writes date values in the form yyyy.mm.dd.	**2018.12.31**

Table 2. Selected Built-In Date Formats

Note that some of the built-in formats listed in Table 2 behave differently depending on the length specified. For example, the WORDDATE format will use the three-letter month abbreviation when a length is specified that is not sufficient for inclusion of the full month name. For more details, consult the specific documentation for each format in *SAS 9.4 Formats and Informats: Reference*.

Built-In Time Formats

Table 3 lists selected built-in time formats. The output shows how the time value corresponding with 15 seconds after 1:14 PM (SAS time 47655) would appear when the format is applied.

Format	Description	Output
`time5.`	Writes time values in the form hh:mm.	`13:14`
`time8.`	Writes time values in the form hh:mm:ss.	`13:14:15`
`time11.2`	Writes time values in the form hh:mm:ss.ss.	`13:14:15.00`
`timeampm8.`	Writes time values in the form hh:mm with AM or PM.	`1:14 PM`
`timeampm11.`	Writes time values in the form hh:mm:ss with AM or PM.	`1:14:15 PM`

Table 3. Selected Built-In Time Formats

Built-In Datetime Formats

Table 4 lists a few of the many built-in datetime formats. The output shows how the datetime value corresponding with 15 seconds after 1:14 PM on December 31, 2018 (SAS datetime 1,861,881,255) would appear when the format is applied. Notice the TIMEAMPM format, which was used above to format a time value, can also be used to format the time portion of a datetime value.

Format	Description	Output
`dateampm.`	Writes datetime values in the form ddmmmyy:hh:mm:ss with AM or PM.	`31DEC18:01:14:15 PM`
`datetime18.`	Writes datetime values in the form ddmmmyy:hh:mm:ss.	`31DEC18:13:14:15`
`datetime20.`	Writes datetime values in the form ddmmmyyyy:hh:mm:ss.	`31DEC2018:13:14:15`
`e8601dt.`	Writes datetime values using ISO 8601 extended notation yyyy-mm-ddThh:mm:ss.ffffff.	`2018-12-31T13:14:15`
`mdyampm25.`	Writes datetime values in the form mm/dd/yyyy hh:mm with AM or PM.	`12/31/2018  1:14 PM`
`timeampm11.`	Writes the time portion of datetime values in the form hh:mm:ss with AM or PM.	`1:14:15 PM`

Table 4. Selected Built-In Datetime Formats

PICTURE FORMATS

If one wishes to express a date, time, or datetime in a manner not conforming to one of the built-in formats, a picture format may be the solution. Picture formats are created using the PICTURE statement in the FORMAT procedure (Base SAS 9.4 Procedures Guide, pp. 959-971).

The PICTURE statement expects a quoted string defining a template for formatting a numeric value. When that numeric value happens to be a date, time, or datetime value, a set of special character codes referred to as directives are available to specify various date and time elements as part of the picture definition. To use these directives, the DATATYPE= option must be included on the PICTURE statement with a value of DATE, TIME, or DATETIME to indicate the specific type of value being formatted.

Selected directives are listed in Table 5. A complete list is available in *Base SAS 9.4 Procedures Guide, Seventh Edition*. Note that the directives are case-sensitive.

Directive	Description	Directive	Description
%a	Short weekday name (e.g. “Mon”)	%q	Quarter of year, number (1-4)
%A	Full weekday name (e.g. “Monday”)	%Q	Quarter of year (e.g. “Quarter4”)
%b	Short month name (e.g. “Dec”)	%s	Fractional seconds
%B	Full month name (e.g. “December”)	%S	Seconds*
%d	Day of the month (1-31)*	%u	Day of week (1-7, Sunday=7)*
%H	Hour, 24-hour clock (0-23)*	%U	Week number (0-53, by Mondays)*
%I	Hour, 12-hour clock (1-12)*	%w	Day of week (0-6, Sunday=0)*
%j	Day of year, number (1-366)*	%W	Week number (0-53, by Sundays)*
%m	Month as a number (1-12)*	%y	Year, two-digit*
%M	Minute (0-59)*	%Y	Year, four-digit
%p	AM or PM	%%	Escape code for % character
*Insert a zero immediately after the percent sign to indicate leading zeroes should be included.			

Table 5. Selected Date and Time Directives for Picture Formats

For example, we could use the following code to create a picture format called “mypic”.

```
proc format;
  picture mypic
    low - high = '%Y-%b-%0d (W%W:D%w)' (datatype=date);
run;
```

Using the put statement shown below to apply this format to the date value of December 31, 2018, results in the output shown.

```
mydate = "31DEC2018"d;
put mydate mypic20.;
```

Output:

```
2018-Dec-31 (W53:D2)
```

CUSTOM FORMATS

Another way to accomplish custom formatting of dates is by assigning specific formatted values to individual dates or date ranges. This is done in PROC FORMAT using the VALUE statement as follows:

```
proc format;
      value potus
            "30APR1789"d - "03MAR1797"d = 'George Washington'
            "04MAR1797"d - "03MAR1801"d = 'John Adams'
            "04MAR1801"d - "03MAR1809"d = 'Thomas Jefferson'
            "04MAR1809"d - "03MAR1817"d = 'James Madison'   …
            ...
      ;
run;
```

When this format is applied, date values will be formatted based on the first range into which the date falls.

```
mydate = "01JAN1800"d;
put "On" mydate worddate. ", the U.S. president was: " mydate potus.;
```

Output:

```
On   January 1, 1800, the U.S. president was: John Adams
```

READING DATES AND TIMES WITH SAS INFORMATS

We have seen how formats allow us to present numerical date and time information in a variety of ways, but sometimes we need to do the opposite. That is, we may have formatted text that incorporates date and/or time information, and we wish to store this information in a SAS data set as date, time, or datetime values. For this task, we turn to informats.

Informats allow us to apply a specific pattern or set of instructions when reading data as input (SAS 9.4 Formats and Informats, p. 227). As was the case with formats, SAS provides a large library of built-in informats as well as the ability to create our own. Table 6 describes several common informats used for reading date and time information.

Informat	Description
DATE.	Reads dates in the form ddmmmyy or ddmmmyyyy, with or without delimiters.
DATETIME.	Reads datetimes in the form ddmmmyy hh:mm:ss.ss or ddmmmyyyy hh:mm:ss.ss, with or without delimiters.
DDMMYY.	Reads dates in the form ddmmyy or ddmmyyyy, with or without delimiters.
E8601DA.	Reads dates specified in the ISO 8601 extended notation yyyy-mm-dd.
E8601DT.	Reads datetimes specified in the ISO 8601 extended notation yyyy-mm-ddThh:mm:ss.ffffff.
MMDDYY.	Reads dates in the form mmddyy or mmddyyyy, with or without delimiters.
TIME.	Reads times in the form hh:mm:ss.ss with optional AM or PM.
YYMMDD.	Reads dates in the form yymmdd or yyyymmdd, with or without delimiters.

Table 6. Selected Informats for Reading Dates, Times, and Datetimes

Informats are typically used in conjunction with the INPUT statement or the INPUT family of functions. This is frequently done when data are being read from an external file. It could also be that a character variable containing date and time information needs to be converted to a numeric date, time, or datetime value.

Several of the informats listed in Table 6 provide some degree of flexibility in the exact formatting of the text they read. For example, the TIME. informat will work with colons separating the hours, minutes, and seconds, but will also work with periods, slashes, dashes, and other characters. It also does not care whether “PM” is uppercase or lowercase.

For the ultimate in flexibility, SAS includes a set of three “ANY” informats that will correctly interpret a wide variety of formatted date and time information. Wicklin (2016) characterizes these informats as “super format[s]” because they can perform the job of several older formats. These are described below in Table 7.

Informat	Description
ANYDTDTE.	Reads dates from various date, time, and datetime forms.
ANYDTDTM.	Reads datetimes from various date, time, and datetime forms.
ANYDTTME.	Reads times from various date, time, and datetime forms.

Table 7. The "ANY" Date and Time Informats

For example, applying the ANYDTDTE20. informat will correctly interpret each of the following text strings (and many others) as representing December 31, 2018:

```
31DEC2018              12/31/2018             12.31.18
2018-12-31             December 31, 2018      Dec. 31, 2018
2018 Dec 31            31-Dec-2018            20181231
```

WORKING WITH SAS DATE AND TIME FUNCTIONS

Over 500 functions are included with SAS 9.4, dozens of which are related to working with dates and times. These functions provide the SAS programmer a rich and robust set of tools for manipulating date and time information. In this paper, we will review several of the most widely used date and time functions. For an exhaustive list, refer to *SAS Functions and CALL Routines: Reference.*

FUNCTIONS FOR GETTING THE CURRENT DATE AND TIME

There are several functions available that can provide information about the current date and/or time at the time of execution:

- DATE – Returns the current day as a SAS date value
- TODAY – An alias for the DATE function
- TIME – Returns the current time of day as a SAS time value
- DATETIME – Returns the current date and time of day as a SAS datetime value

None of these functions need any arguments. Table 8 shows what values each function would return if they were run at 1:00pm on May 22, 2018. The return values are shown both as unformatted numerical values and as formatted values to better illustrate the functionality.

FUNCTION CALL	RETURN VALUE (UNFORMATTED)	RETURN VALUE (FORMATTED)
DATE()	**21326**	**22MAY2018**
TODAY()	**21326**	**22MAY2018**
TIME()	**46800**	**13:00:00**
DATETIME()	**1842613200**	**22MAY2018:13:00:00**

Table 8. Example Function Calls - Getting the Current Date and Time

FUNCTIONS FOR EXTRACTING COMPONENTS OF THE DATE OR TIME

There can be a lot of information packed into a date, time, or datetime value. Sometimes, we wish to extract a part of that information to use separately. Several functions facilitate this extraction.

Deconstructing a Datetime Value

- DATEPART – Extracts the date from a SAS datetime value
- TIMEPART – Extracts the time from a SAS datetime value

Both functions accept a SAS datetime value as an argument. As previously discussed, a datetime value represents a specific time of day on a specific date. The DATEPART and TIMEPART functions,

respectively, return that specific date as a SAS date value and that specific time of day as a SAS time value. Table 9 shows the values each function would return if they were run at 1:00pm on May 22, 2018.

FUNCTION CALL	RETURN VALUE (UNFORMATTED)	RETURN VALUE (FORMATTED)
`DATEPART(DATETIME())`	`21326`	`22MAY2018`
`TIMEPART(DATETIME())`	`46800`	`13:00:00`

Table 9. Example Function Calls - Deconstructing a Datetime Value

Deconstructing a Date Value

- YEAR – Extracts the year from a SAS date value
- MONTH – Extracts the month from a SAS date value
- DAY – Extracts the day of the month from a SAS date value
- WEEKDAY – Returns an integer corresponding to the day of the week of a SAS date value
- WEEK – Returns an integer corresponding to the week number of the year of a SAS date value

The YEAR, MONTH, and DAY functions accept a date value as an argument and return the corresponding year, month, or day, respectively. The WEEKDAY function also accepts a date value as an argument, but it returns an integer 1 through 7 representing the corresponding day of the week. Sunday is represented by the number 1 while 7 is associated with Saturday.

The WEEK function is somewhat more complex. It accepts a SAS date value and returns the corresponding week number within the year (0 through 53), but it can count weeks 3 different ways. Its second argument is a descriptor indicating the counting method, which can have the value “U”, “V”, or “W”.

The default, “U”, indicates Week 1 begins on the first Sunday of the year, while “W” specifies Week 1 starts with the first Monday of the year. In both cases, dates prior to the beginning of Week 1 are considered part of Week 0.

The “V” descriptor is somewhat different. It indicates weeks begin on Mondays, and Week 1 is the week including both January 4 and the first Thursday of the year. In this scheme, dates prior to Week 1 are considered part of the last week of the preceding year.

Table 10 provides examples of the values each function would return if run at 1:00pm on May 22, 2018.

FUNCTION CALL	RETURN VALUE
`YEAR(TODAY())`	`2018`
`MONTH(TODAY())`	`5`
`DAY(TODAY())`	`22`
`WEEKDAY(TODAY())`	`3`
`WEEK(TODAY())`	`20`
`WEEKDAY("01JAN2017"d)`	`1`
`WEEK("01JAN2017"d,"U")`	`1`
`WEEK("01JAN2017"d,"V")`	`52`
`WEEK("01JAN2017"d,"W")`	`0`

Table 10. Example Function Calls - Deconstructing a Date Value

Deconstructing a Time Value

- HOUR – Extracts the hour from a SAS time or datetime
- MINUTE – Extracts the minute from a SAS time or datetime
- SECOND – Extracts the second from a SAS time or datetime

The HOUR, MINUTE, and SECOND functions can accept either a SAS time value or a SAS datetime value. In either case, they return the corresponding hour, minute, or second, respectively. Table 11 shows the values each function would return if run at 1:00pm.

FUNCTION CALL	RETURN VALUE
`HOUR(TIME())`	`13`
`MINUTE(TIME())`	`0`
`SECOND(TIME())`	`0`

Table 11. Example Function Calls - Deconstructing a Time Value

FUNCTIONS FOR BUILDING DATES AND TIMES FROM THEIR COMPONENTS

The functions in the previous section allowed us to break apart a date, time, or datetime value into its constituent pieces. We turn next to the opposite task. Below are three functions for constructing a date, time, or datetime value from those individual elements.

- MDY – Constructs a SAS date value from month, day, and year values
- HMS – Constructs a SAS time value from hour, minute, and second values
- DHMS – Constructs a SAS datetime value from date, hour, minute, and second values

The MDY function accepts three numerical arguments: a month, a day of the month, and a year, in that order. It returns a SAS date value constructed from those three elements. Similarly, the HMS function accepts numerical arguments containing the hour, minute, and second and returns the corresponding SAS time value.

The DHMS function allows us to assemble a SAS datetime value. As arguments, it takes a SAS date value followed by an hour, minute, and second. Alternatively, a SAS time value can be passed in as the number of seconds (fourth argument), in which case the hours and minutes (second and third arguments) should both be zero.

Notice that the DHMS function does not directly allow us to produce a datetime value from a year, month, day, hour, minute, and second. However, this could be accomplished by simply nesting the MDY function in as the first argument to the DHMS function.

Table 12 contains example calls to each of these functions along with the values which would be returned if executed at 1:00pm on May 22, 2018.

FUNCTION CALL	RETURN VALUE (UNFORMATTED)	RETURN VALUE (FORMATTED)
`MDY(5,22,2018)`	`21326`	`22MAY2018`
`HMS(13,0,0)`	`46800`	`13:00:00`
`DHMS(TODAY(),13,0,0)`	`1842613200`	`22MAY2018:13:00:00`
`DHMS(MDY(5,22,2018),13,0,0)`	`1842613200`	`22MAY2018:13:00:00`
`DHMS(DATE(),0,0,TIME())`	`1842613200`	`22MAY2018:13:00:00`

Table 12. Example Function Calls - Constructing Date, Time, and Datetime Values

FUNCTIONS DEALING WITH TIME INTERVALS

The functions we've looked at so far deal with individual points in time. However, there may be times when we are interested in the interval spanning between two points in time. Since SAS dates are represented by numbers, counting the number of days from one date to another can be as simple as subtracting the two date values. However, for more complex tasks, SAS gives us two powerful functions for working with time intervals: INTCK and INTNX.

- INTCK - Returns the number of interval boundaries of a given kind that lie between two dates, times, or datetime values.
- INTNX - Increments a date, time, or datetime value by a given time interval, and returns a date, time, or datetime value.

The INTCK Function

The INTCK function allows us to count several different types of built-in intervals. Common types include DAY, WEEK, MONTH, QTR, YEAR when working with date values and HOUR, MINUTE, and SECOND when working with time values. The latter can also be used with datetime values. To use a date interval type with datetime values, a prefix of DT must be added to the interval type (e.g. DTDAY, DTWEEK, etc.). For a complete list of interval types, refer to *SAS 9.4 Language Reference: Concepts* (pp. 130-140).

There are two different methods for counting the number of interval boundaries. The default method, called the discrete method, counts interval boundaries based on fixed points on the calendar or clock (*SAS 9.4 Functions*, p. 29). For example, using the discrete method, the number of years from December 31, 2017, to January 1, 2018, is one because there is exactly one yearly boundary between those two dates (that is, the boundary between calendar years 2017 and 2018). The number of years between January 1, 2017, and December 31, 2018, is also one because, again, there is exactly one yearly boundary between those two dates.

The other method for counting interval boundaries is the continuous method. This method may be a bit more intuitive because it's the way we commonly think of time intervals. Using the continuous method, the number of years from December 31, 2017, to January 1, 2018, is zero. That's because under this method the first yearly interval begins on the start date, December 31, 2017. The end date, January 1, 2018, occurs within that first interval so no interval boundary is spanned. On the other hand, the number of years between January 1, 2017, and December 31, 2018, is still just one. The first yearly boundary is crossed on January 1, 2018, one year after the interval begins, but a second yearly boundary is not reached before the end date.

The examples above and several others are presented below in Table 13. Detailed syntax and options can be found in *SAS 9.4 Functions and CALL Routines: Reference* (pp. 699-705). Morgan (2015) discusses some advanced features of the INTCK function such as shifting discrete intervals to start at other times and defining non-standard intervals.

FUNCTION CALL	RETURN VALUE	COMMENTS
`intck('YEAR',"31DEC2017"d, "01JAN2018"d,'D')`	1	A discrete year boundary occurs between calendar years 2017 and 2018.
`intck('YEAR',"01JAN2017"d, "31DEC2018"d,'D')`	1	Same as previous.
`intck('YEAR',"31DEC2017"d, "01JAN2018"d,'C')`	0	It's less than one continuous year from the start date to the end date.
`intck('YEAR',"01JAN2017"d, "31DEC2018"d,'C')`	1	It's more than one but less than two continuous years from the start date to the end date.
`intck('YEAR',"29FEB2016"d, "28FEB2017"d,'C')`	1	The year interval beginning February 29 of a leap year ends on the following February 28.
`intck('YEAR',"29FEB2016"d, "28FEB2020"d,'C')`	3	However, if the end date is also in a leap year, the yearly boundary occurs February 29.
`intck('MONTH',"01FEB2017"d, "01MAR2017"d,'C')`	1	The length of a continuous month interval depends when it started. Here, 29 days makes 1 month.
`intck('MONTH',"01MAR2017"d, "31MAR2017"d,'C')`	0	Here, on the other hand, 31 days does not make an entire month.
`intck('MONTH',"29JAN2017"d, "28FEB2017"d,'C')`	1	Here, 30 days makes a month since there can be no February 29.
`intck('MONTH',"29JAN2016"d, "28FEB2016"d,'C')`	0	However, the same 30-day span in a leap year does not constitute a month.
`intck('HOUR',"05:45"t, "06:15"t,'D')`	1	A discrete hourly boundary occurs at 06:00.
`intck('HOUR',"05:45"t, "06:15"t,'C')`	0	It's less than one continuous hour from 05:45 to 06:15.
`intck('HOUR',"31DEC2017:23:00"dt, "01JAN2018:01:00"dt,'C')`	2	It's two continuous hours from 11pm one day to 1am the next.
`intck('DTDAY',"31DEC2017:23:00"dt, "01JAN2018:01:00"dt,'D')`	1	There's one discrete day boundary between Dec. 31 and Jan. 1, regardless of the time of day.

Table 13. Example Function Calls – INTCK

The INTNX Function

Whereas the INTCK function allows us to count interval boundaries between two points in time, the INTNX function allows us to specify the starting point and number of intervals to derive the ending timepoint. As with INTCK, the first argument is the interval type and the second argument is the starting date, time, or datetime. The third argument specifies the number of intervals by which to increment the value of the second argument.

An optional fourth argument referred to as the "alignment" allows to specify the point within the interval to return. The default value is "BEGINNING" (or "B"), but the function will also accept "MIDDLE" ("M"), "END" ("E"), or "SAME" ("S"). A value of "SAME" indicates that the date, time, or datetime being returned should have the same alignment in its interval as the starting value had within its respective interval.

Some basic examples are provided in Table 14. Detailed syntax and options can be found in *SAS 9.4 Functions and CALL Routines: Reference* (pp. 723-730). In addition, refer to Wicklin (2017) for a method that uses the INTCK and INTNX functions together to compute the number of years and days between two dates (e.g. 3 years, 12 days).

FUNCTION CALL	RETURN VALUE (FORMATTED)	COMMENTS
`intnx('DAY',"01JUN2018"d,60)`	31JUL2018	60 days after June 1 is July 31.
`intnx('MONTH', "15JUN2018"d,18)`	01DEC2019	18 months after June 2018 is Dec. 2019. By default, INTNX returns the beginning of that interval, Dec. 1.
`intnx('MONTH', "15JUN2018"d,18,'E')`	31DEC2019	Same as above, but the alignment option specifies the end of the interval, Dec. 31.
`intnx('MINUTE', "05:30"t,180)`	08:30	180 minutes after 05:30 is 08:30.
`intnx('DTYEAR', "01SEP2017:14:00"dt, 1,'S')`	01SEP2018: 14:00:00	Increment the date by one year using the "S" alignment option to specify the same position within the year.

Table 14. Example Function Calls - INTNX

A FUNCTION FOR THE HOLIDAYS

- HOLIDAY - Returns a SAS date value of a specified holiday for a specified year.

Processing which is date-dependent may need to take holidays into account. SAS provides the HOLIDAY function for computing the dates of various holidays. The HOLIDAY function takes two arguments, a holiday keyword and a numerical year, and returns a SAS date value.

Holiday keywords include CHRISTMAS, THANKSGIVING, EASTER, FATHERS and MOTHERS, LABOR, MLK, MEMORIAL, VETERANS, and more. A complete list of keywords can be found in *SAS 9.4 Functions and CALL Routines: Reference* (pp. 654-655). Selected examples are shown in Table 15.

FUNCTION CALL	RETURN VALUE (FORMATTED)
`holiday('THANKSGIVING',2018)`	22NOV2018
`holiday('MOTHERS',2019)`	12MAY2019
`holiday('EASTER',2019)`	21APR2019

Table 15. Example Function Calls - HOLIDAY

DATE AND TIME MACRO VARIABLES

When a new SAS session is started, dozens of macro variables are automatically created. Several of these having to do with the current date and time are listed in Table 16 (*SAS Macro Language* Reference, pp. 24-28). It can be convenient to use these macro variables in titles, footnotes, or elsewhere as desired to place a datestamp and/or timestamp within a report or other output.

One must keep in mind that these automatic macro variables reflect the date and time at the moment a SAS session begins. They are not updated during the life of the session. In contrast, the DATE and TIME functions discussed earlier return values based on the moment they are executed.

MACRO VARIABLE	DESCRIPTION	SAMPLE VALUE
SYSDATE	Date the SAS session began executing (two-digit year)	01JAN18
SYSDATE9	Date the SAS session began executing (four-digit year)	01JAN2018
SYSDAY	Day of the week the SAS session began executing	Monday
SYSTIME	Time of day the SAS session began executing	17:30

Table 16. Selected Automatic Macro Variables

CONCLUSION

The SAS programmer has many tools available for reading, formatting, manipulating, analyzing, and reporting information about dates and times. These include an extensive library of built-in formats and informats, mechanisms for building custom formats, a vast array of functions, and several automatic macro variables. This paper offers only a high-level overview of these tools. The included references and other recommended reading provide a more comprehensive treatment of the subject.

REFERENCES

Base SAS® 9.4 Procedures Guide, Seventh Edition. Cary, NC: SAS Institute Inc., 2017 https://documentation.sas.com/api/docsets/proc/9.4/content/proc.pdf.

Morgan, Derek P. "Demystifying Date and Time Intervals." Proceedings of the SAS Global Forum 2015 Conference. Cary, NC: SAS Institute Inc., 2015. https://support.sas.com/resources/papers/proceedings15/2460-2015.pdf.

SAS® 9.4 Formats and Informats: Reference. Cary, NC: SAS Institute Inc., 2013. https://support.sas.com/documentation/cdl/en/leforinforref/64790/PDF/default/leforinforref.pdf.

SAS® 9.4 Functions and CALL Routines: Reference, Fifth Edition. Cary, NC: SAS Institute Inc., 2016. https://documentation.sas.com/api/docsets/lefunctionsref/9.4/content/lefunctionsref.pdf.

SAS® 9.4 Language Reference: Concepts, Sixth Edition. Cary, NC: SAS Institute Inc., 2016. https://documentation.sas.com/api/docsets/lrcon/9.4/content/lrcon.pdf.

SAS® 9.4 Macro Language Reference, Fifth Edition. Cary, NC: SAS Institute Inc., 2016. https://documentation.sas.com/api/docsets/mcrolref/9.4/content/mcrolref.pdf.

SAS® 9.4 System Options: Reference, Fifth Edition. Cary, NC: SAS Institute Inc., 2016. https://documentation.sas.com/api/docsets/lesysoptsref/9.4/content/lesysoptsref.pdf.

Wicklin, Rick. "One informat to rule them all: Read any date into SAS." The DO Loop blog, SAS Institute Inc., 11 Nov 2016. https://blogs.sas.com/content/iml/2016/11/11/anydtdte-informat-read-any-date-sas.html.

Wicklin, Rick. "INTCK and INTNX: Two essential functions for computing intervals between dates in SAS." The DO Loop blog, SAS Institute Inc., 15 May 2017. https://blogs.sas.com/content/iml/2017/05/15/intck-intnx-intervals-sas.html.

RECOMMENDED READING

Morgan, Derek P. *The Essential Guide to SAS® Dates and Times, Second Edition*. Cary, NC: SAS Institute Inc., 2014.

CONTACT INFORMATION

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

Joshua M. Horstman
Nested Loop Consulting
317-721-1009
josh@nestedloopconsulting.com