How to display values in time format in SAS?

Authored by

stats writer

November 19, 2025

Introduction to Time Formats in SAS

Handling time data accurately and presenting it legibly is a fundamental task in any data analysis environment. In SAS, the preferred method for displaying raw numeric time values in a human-readable format is through the application of a time format, particularly the `TIMEw.` format. This powerful formatting tool allows analysts to transform a stored numeric measure--representing time--into a familiar display showing hours, minutes, and seconds. Understanding how TIMEw. format operates is crucial for anyone working with temporal data structures within the SAS environment.

The `TIMEw.` format structures the output specifically in the standard HH:MM:SS representation. Here, HH signifies the hour, MM represents the minute, and SS denotes the second. The 'w' component specifies the width of the output field, controlling how many characters are displayed. Importantly, when SAS applies this format, it converts the underlying numeric data into a displayable character string based on the defined time structure. This process is essential because while calculations are performed on the raw numeric time value, presentation requires structured text.

It is vital to recognize that SAS stores time values internally as a single numeric value, specifically the count of seconds elapsed since midnight (00:00:00) of that day. For instance, if a duration is 7:30:00, SAS stores the corresponding total number of seconds (27,000 seconds). The `TIMEw.` format is highly versatile, capable of handling both variables that are explicitly stored as numeric time measures and those that might be derived or manipulated as character representations, ensuring consistent display across different data types, provided the underlying value correctly represents seconds since midnight.

Understanding SAS Time Values: Seconds Since Midnight

To effectively format time data in SAS, one must grasp the core principle of its internal storage mechanism. Unlike systems that might store time as a simple text string or complex object, SAS standardizes time measures by calculating the total elapsed seconds from the start of the day (midnight). This numeric standardization allows for robust mathematical operations, such as calculating differences between time points or summing durations, without the complexities inherent in text-based date/time calculations.

This numeric representation means that a value of 0 corresponds to 00:00:00, and the maximum value for a 24-hour cycle is 86,399, corresponding to 23:59:59 (since there are 86,400 seconds in a full day, 86,400 seconds would represent the next midnight, typically stored as day 1, time 0, depending on the variable definition). When an input statement reads time data in a recognizable format (like HH:MM:SS), the TIMEw. format acts as an informat to convert the textual input into this standardized numeric count of seconds.

This approach ensures high precision. When we apply formats like `TIME8.`, we are instructing SAS to take that raw second count and project it back into the conventional clock display (HH:MM:SS). Without an explicit format, SAS would simply print the large numeric integer (e.g., 27000), which is meaningless to most users. Therefore, formatting is a presentation layer necessity, not a data storage alteration.

Applying Core Functions for Time Formatting

Consider a practical scenario where you have a variable within your data set called **duration** that holds a time value corresponding to **7:30:00**. Although this value is stored internally as seconds (27,000), SAS provides several functions--most notably the `PUT` function--to manipulate and format this time value into various display formats required for reporting or further analysis. The `PUT` function is especially versatile as it converts a numeric value (like the duration in seconds) into a character string using a specified format.

Below, we explore the primary SAS functions available for manipulating and formatting time values, showcasing the results based on our example duration of 7:30:00. These functions demonstrate not only how to display the full time string but also how to extract specific components like hours or minutes.

PUT(duration, time8.) - This function formats the duration value using the `TIME8.` format. This structure is ideal for standard display, ensuring the output includes hours, minutes, and seconds, with a total field length of 8 characters (including colons).

This operation will produce the output: 7:30:00.

PUT(duration, hhmm.) - This is a specialized time format that suppresses the display of seconds, focusing solely on hours and minutes. It is often preferred in reports where second-level precision is unnecessary.

This operation will produce the output: 7:30.

PUT(duration, hours5.2) - This specific format converts the duration into decimal hours. The `5.2` component indicates a total width of 5 characters, with 2 digits reserved for the decimal component. This is highly useful for calculations or presentations requiring fractional hour representations.

This operation will produce the output: 7.50. (Since 30 minutes is 0.5 hours).

hour(duration) - Unlike the `PUT` function which formats the entire value, the `HOUR` function extracts and returns only the integer portion representing the hour component of the time value.

This operation will produce the output: 7

minute(duration) - Similarly, the `MINUTE` function isolates and returns the integer portion representing the minute component of the time value.

This operation will produce the output: 30.

second(duration) - This function extracts and returns the integer portion representing the second component of the time value.

This operation will produce the output: 0.

The following comprehensive example illustrates how to apply these distinct formatting techniques within a SAS data step to produce a derived data set containing all these different time representations side-by-side.

Example: Display Values in Time Formats in SAS

Practical Application: Setting Up the Dataset

To demonstrate the application of these formats, we will begin by creating a sample data set. This data set, named `my_data`, contains information about the duration it took various athletes to complete a specific task. We use the `TIME8.` informat within the `input` statement to ensure the textual time inputs are correctly converted and stored internally as numeric seconds since midnight.

Note the structure of the `datalines` statement: the durations are entered in the standard HH:MM:SS format, and the `time8.` informat handles the conversion. This initial step is critical for setting up the time data correctly before any output formatting can occur.

```
/*create dataset*/
data my_data;
input athlete $ duration time8.;
datalines;
A 04:15:00
B 10:09:15
C 7:30:00
D 18:55:00
E 14:23:59
F 23:45:10
;
```

run;

/*view dataset*/
proc print data=my_data;

When the initial data set `my_data` is viewed using `PROC PRINT` without any explicit formatting applied to the `duration` variable, SAS defaults to displaying the raw internal numeric value. This clearly illustrates how time is stored: as a large integer representing the total seconds since midnight.

Obs	athlete	duration
1	A	15300
2	B	36555
3	C	27000
4	D	68100
5	E	51839
6	F	85510

The output above confirms that the values are stored as seconds. For instance, the first observation (Athlete A, 04:15:00) is equivalent to 15,300 seconds (4 hours * 3600 seconds/hour + 15 minutes * 60 seconds/minute). This internal representation is necessary for accurate calculations. It is worth reiterating a key fact for time calculations: there are precisely 86,400 seconds in a standard 24-hour day.

Transforming Numeric Time into Diverse Formats

The primary goal is now to take these raw numeric duration values and transform them into various easily digestible time formats suitable for analysis and reporting. We achieve this by creating a new data set, `new_data`, where we use the `PUT` function (for format conversion to a character string) and the specific time extraction functions (`HOUR`, `MINUTE`, `SECOND`) to generate seven new columns based on the original `duration` variable.

Each new variable showcases a different technique for presenting the underlying time data. Variables prefixed with `duration_` followed by the format name (e.g., `duration_time8`) are created using the `PUT` function along with the relevant format specification (`TIME8.`, `HHMM.`, `HOUR5.2`). The extraction functions (`HOUR`, `MINUTE`, `SECOND`) directly yield numeric values representing those specific components.

```
/*create new dataset with duration printed in various time formats*/
data new_data;
set my_data;
duration_time8 = put(duration, time8.);
duration_hhmm = put(duration, hhmm.);
duration_hour52 = put(duration, hour5.2);
duration_hour = hour(duration);
duration_minute = minute(duration);
duration_second = second(duration);
run;

/*view new dataset*/
proc print data=new_data;
```

This code block efficiently leverages SAS functions to generate a comprehensive output detailing every facet of time formatting discussed. Running `PROC PRINT` on `new_data` allows us to compare the resulting column formats directly.

Interpreting the Formatted Output Columns

Upon execution of the data step and the print procedure, we obtain a new table where the original numeric duration is complemented by six new variables, each representing a distinct formatted version of the time data. Analyzing this output is essential for understanding the utility of each format type.

Obs	athlete	duration	duration_time8	duration_hhmm	duration_hour52	duration_hour	duration_minute	duration_second
1	A	15300	4:15:00	4:15	4.25	4	15	0
2	B	36555	10:09:15	10:09	10.15	10	9	15
3	C	27000	7:30:00	7:30	7.50	7	30	0
4	D	68100	18:55:00	18:55	18.92	18	55	0
5	E	51839	14:23:59	14:24	14.40	14	23	59
6	F	85510	23:45:10	23:45	23.75	23	45	10

Reviewing the resulting table, observe how each new column manipulates the presentation of the time value:

duration_time8: This column applies the standard `TIME8.` format. It consistently displays the duration in the full **HH:MM:SS** structure, which is the most comprehensive clock display format.
duration_hhmm: Utilizing the `HHMM.` format, this column truncates the seconds display, presenting only **hours and minutes** (HH:MM). This simplification is highly beneficial for high-level reporting

where seconds are noise.
duration_hour52: This column displays the time as **decimal hours** (e.g., 7.50 for 7 hours and 30 minutes). This is a character representation derived using the `HOURd.f` format, making time directly comparable to other fractional numeric data.
duration_hour: This variable, created by the `HOUR()` function, contains only the integer value representing the **hour** component (0-23).
duration_minute: This variable, created by the `MINUTE()` function, contains only the integer value representing the **minute** component (0-59).
duration_second: This variable, created by the `SECOND()` function, contains only the integer value representing the **seconds** component (0-59).

These distinct columns illustrate the dual capability of SAS: using formats via the `PUT` function for aesthetic display (creating character variables) and using extraction functions (`HOUR`, `MINUTE`, `SECOND`) for isolating specific numeric value components for further mathematical analysis or grouping.

Selecting the Appropriate Time Display Format

The choice of which time format to employ is entirely dependent on the analytical or reporting objective. If the goal is high-precision tracking, the `TIME8.` format is indispensable. If the analysis requires combining time units with standard numerical calculations--such as averaging productivity per fractional hour--the `HOURd.f` format (generating `duration_hour52`) is the most efficient.

Conversely, if the requirement is merely to categorize observations based on the hour of the day or to check if a specific event occurred within a certain minute, the extraction functions (`HOUR`, `MINUTE`, `SECOND`) provide the fastest way to obtain the necessary numeric components without having to manipulate a full time string.

By mastering the application of the TIMEw. format family and the corresponding extraction functions, data professionals can ensure that temporal data in SAS data sets is always presented clearly, accurately, and in the most suitable format for the intended audience or calculation. Feel free to incorporate whichever format best addresses your specific data visualization and analytical needs.

The following tutorials explain how to perform other common tasks in SAS: