How to Easily Convert Character Variables to Date Variables in SAS

Authored by

stats writer

December 1, 2025

Working effectively with time-series data or transactional records in SAS requires accurate handling of data types, especially when dealing with dates. A frequent challenge for data analysts is converting dates stored as text--a character variable--into a numeric date variable that SAS can recognize and calculate.

The primary and most robust mechanism for achieving this critical transformation in SAS is through the use of the INPUT function, coupled with the precise specification of a date Informat. An Informat dictates how SAS should read raw data, ensuring that the character string, such as '01012022', is interpreted correctly according to a specified pattern, like MMDDYY. Once successfully read using the INPUT function, the resulting value is stored internally as a date variable--a numeric value representing the number of days since January 1, 1960. This internal numeric representation is essential for performing sophisticated date arithmetic and manipulation.

To initiate the conversion process, we must leverage the fundamental power of the INPUT function. This function is specifically designed to read a source value using a specified Informat and then return the value using the corresponding data type. When converting a date stored as text, the INPUT function is indispensable, allowing us to dictate the exact structure of the incoming text so that SAS does not misinterpret the month, day, or year components. A critical second step is applying a FORMAT statement immediately after conversion, which tells SAS how to display the newly created numeric date value in a readable format for reporting and visualization.

The standard syntax for this transformation involves creating a new target variable, assigning it the output of the **input()** function, and then ensuring the appropriate FORMAT is applied to that new variable. This two-part approach--conversion followed by formatting--is foundational to effective date management in SAS programming, ensuring both computational accuracy and presentation clarity.

```
date_var = input(character_var, MMDDYY10.);
format date_var MMDDYY10.;
```

The following detailed example illustrates the application of this function in a practical data scenario, transforming raw sales data stored as character strings into actionable date variables.

Understanding Data Types and Date Representation in SAS

Before diving into the conversion mechanics, it is essential to appreciate how SAS handles fundamental data types, specifically the distinction between character strings and numeric date values. A character variable is simply a sequence of letters, numbers, or symbols, stored exactly as input, regardless of whether that string looks like a date or a number. Because it is stored as text, SAS cannot perform arithmetic operations on it, meaning you cannot calculate the difference

between '01012022' and '01052022' using standard date functions. This is why conversion is necessary.

Conversely, a date variable in SAS is stored as a standard numeric value. This number represents the count of days elapsed since the reference date of January 1, 1960. For example, January 2, 1960, is stored internally as the number 1, and January 1, 2022, is a much larger numeric integer. This internal standardization allows SAS to handle complex temporal calculations efficiently, such as determining the number of working days between two points or aggregating data by week, month, or quarter. The goal of the conversion process, therefore, is to accurately translate the character string representation into this standardized numeric count.

The crucial bridge between the character representation and the internal numeric date value is the Informat. An Informat acts as a template, guiding the INPUT function to correctly parse the character string. If the source data is structured as day-month-year (DDMMYY), using an Informat designed for month-day-year (MMDDYY) will result in incorrect dates and data corruption. Understanding the exact structure of your raw character data is the first prerequisite for a successful conversion in SAS.

Utilizing the INPUT Function for Conversion

The INPUT function is the cornerstone of data type conversion in SAS, serving the specific purpose of reading a source variable and reinterpreting it based on a user-defined Informat. When applied to date conversion, the function takes two mandatory arguments: the source character variable containing the date string and the specific date Informat that matches the structure of that string. The output of this function is a numeric value that represents the calculated date, ready to be stored as a proper date variable.

Consider a scenario where the source data format is consistent, such as eight digits representing the date without separators (e.g., YYYYMMDD). To convert this accurately, you would use an Informat like YYYYMMDD8. The numerical suffix (8) indicates the total width of the character field being read. If, however, the source date includes separators like slashes or hyphens (e.g., 01/15/2022), you would typically need to use an Informat wide enough to accommodate the separators, such as MMDDYY10., which accounts for the two slashes and the total ten characters. Choosing the precise Informat is arguably the most critical step in preventing data transformation errors.

It is important to understand the flow within the SAS DATA step. The INPUT function creates a new numeric variable based on the transformation. If the target variable name is the same as the source character variable, the old character value will be overwritten and replaced by the new numeric date value, potentially causing confusion if the original character data is needed later. Best practice strongly recommends creating a distinct new variable (e.g., `new_day`) to store the result of

the conversion, thereby preserving the original source data for verification or auditing purposes, as demonstrated in the practical example below.

Practical Demonstration of Character-to-Date Conversion

To illustrate the conversion process, consider a scenario involving raw sales data where the transaction date, labeled `day`, is currently stored as a text string in DDMMYYYY format. This structure prevents direct calculation of metrics like sales velocity or time-based aggregation. We must convert the **day** character variable into a proper date variable.

We begin by creating a sample dataset named `original_data` using the DATALINES statement. Notice that the `day` variable is explicitly defined as a character type by appending the dollar sign ($) in the INPUT function statement within the DATA step, ensuring SAS reads it as text initially. Running a PROC PRINT confirms that `day` is currently non-numeric.

```
/*create dataset*/
data original_data;
input day $ sales;
datalines;
01012022 15
01022022 19
01052022 22
01142022 11
01152022 26
01212022 28
;
run;

/*view dataset*/
proc print data=original_data;
```

Obs	day	sales
1	01012022	15
2	01022022	19
3	01052022	22
4	01142022	11
5	01152022	26
6	01212022	28

As observed in the output table above, the variable **day** is treated as a character variable, despite its numeric appearance. Our objective is to generate a new variable that contains the correct, numerically encoded date representation, which will unlock its utility for further statistical analysis within the SAS environment.

Implementing the Conversion Code and Formatting

The actual conversion takes place within a new DATA step, where we read from the `original_data` dataset and apply the INPUT function. Since the source format is eight digits (DDMMYYYY), we must use an appropriate eight-digit informat. We will adhere to the original code's preference for MMDDYY10., which is highly common for date reading in SAS, assuming the structure is MMDDYY (or DDMMYY) followed by the year.

In the following code block, we create the new date variable called `new_day`. We call the **input** function, passing the source variable `day` and the MMDDYY10. informat. Crucially, we then apply the MMDDYY10. format to the new variable using the FORMAT statement. This action ensures that the underlying numeric value is displayed in a human-readable format, preventing SAS from printing the raw integer count, which would be nonsensical to the end-user.

Additionally, we utilize the DROP statement to exclude the original character variable, `day`, from the final dataset `new_data`. This step is often performed for data cleanliness, ensuring the dataset only contains the correctly typed date column, `new_day`, and reducing redundancy. This complete process--setting, converting, formatting, and dropping--creates a clean, analytically useful dataset.

```
/*create new dataset where 'day' is in date format*/
data new_data;
set original_data;
new_day = input(day, MMDDYY10.);
format new_day MMDDYY10.;
drop day;
run;

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

Obs	sales	new_day
1	15	01/01/2022
2	19	01/02/2022
3	22	01/05/2022
4	11	01/14/2022
5	26	01/15/2022
6	28	01/21/2022

Upon reviewing the output generated by PROC PRINT, we can clearly observe that the new variable, `new_day`, is successfully interpreted and displayed as a conventional date. This confirms the successful use of the INPUT function and the proper application of the date format, transforming the raw text into a fully functional date variable.

Note: The use of the DROP statement is purely for presentation and organization. If you needed to retain the original character string for auditing or troubleshooting purposes, you would simply omit the DROP statement.

Choosing the Right SAS Date Informat

The selection of the appropriate Informat is the single most critical factor determining the success of the character-to-date conversion. SAS offers a wide array of informats to handle virtually any text representation of a date, ranging from standard numeric layouts to highly specific text strings containing month names. Misalignment between the source character format and the specified Informat will result in missing values (represented by dots '.'), incorrect dates, or truncated data, requiring careful inspection of the source data structure.

The most commonly used informats for standard numeric date strings include MMDDYY10., DDMMYY10., YYMMDD10., and DATE9. These informats are designed to handle dates in various sequences (Month/Day/Year, Day/Month/Year, Year/Month/Day) and typically require a width (like 10) to account for separators such as slashes or hyphens. If your source data uses text abbreviations for months, such as 'JAN2022', you would need to use an informat like DATE7. or DATE9., which specifically tells SAS to look for three-letter month abbreviations.

For highly ambiguous or non-standard date formats, developers often resort to using the ANYDTDTE. Informat. This powerful informat attempts to automatically recognize a date based on common patterns and heuristics. While useful for messy data, relying on ANYDTDTE. can sometimes lead to unexpected results if the data is genuinely ambiguous (e.g., is '01/02/03' January 2, 2003, or February 1, 2003?). Therefore, whenever possible, explicitly defining the

informat, such as using MMDDYY10., is considered best practice for consistency and reliability in production environments.

Troubleshooting Common Conversion Errors

Despite the simplicity of the INPUT function, conversion errors are frequent, usually stemming from a mismatch between the character data and the specified Informat. The most common symptom of a failed conversion is the appearance of missing values (the numeric dot '.') in the new date variable. This indicates that SAS attempted to read the character variable but could not interpret it according to the template provided by the informat, often leading to warnings in the SAS log.

One primary source of error is incorrect width specification. If the character string is '01/01/2022' (10 characters), but you specify an informat width of 8, SAS will only attempt to read the first eight characters ('01/01/20') and fail the conversion. Conversely, specifying an overly large width is generally harmless but inefficient. Analysts must rigorously verify the exact length and presence of delimiters in the source data. Another frequent mistake is misinterpreting the sequence; for instance, using DDMMYY. when the data is structured as MMDDYY. This error typically results in the creation of seemingly valid but profoundly incorrect dates (e.g., January 5th is read as May 1st).

To diagnose conversion failures effectively, always examine the SAS log for notes regarding "Invalid data." When troubleshooting, temporarily use the PUT function to output the raw numeric date value of the newly created variable into a temporary character column. If the output is a very large positive integer, the conversion was successful, and the issue lies solely in the FORMAT statement. If the output is a period or a number near zero, the **input()** function failed to read the data correctly, requiring adjustment of the Informat itself.

Advanced Date Handling and Manipulations

Once a date variable has been successfully created, SAS offers powerful functions for temporal manipulation. Because dates are numeric, they can be treated arithmetically. Subtracting one date variable from another yields the number of days between the two dates. This capability is fundamental for calculating key metrics like age, duration of stay, or time elapsed since an event. Furthermore, specific functions like `TODAY()` allow comparison against the current system date, while functions like `DATEPART()` can extract date components from datetime variables.

For more complex operations, SAS provides specialized functions such as `INTNX` and `INTCK`. The `INTNX` function allows you to increment a date by a specified interval (e.g., moving forward one month or three quarters), which is invaluable for forecasting or generating future reporting periods. The `INTCK` function calculates the number of interval boundaries crossed between two dates (e.g., how many months or years are between Date A and Date B). Mastering these tools, which only

work on true date variables, significantly enhances the analytical power of the SAS environment.

Finally, remember that while MMDDYY10. is a common display format, it is only one option. SAS supports numerous date formats to meet diverse presentation requirements, including YEAR4., YYMMDD., and DDMONYY. to name a few. The appropriate format should be chosen based on regional standards and reporting clarity. By correctly converting the initial character variable and utilizing these formatting and manipulation tools, analysts can ensure their data is robust and ready for sophisticated analysis.