

Paper CC-020

Conversion from Daylight Savings to Standard Time

Audrey Ewen, GoodCents, Grayson, GA

ABSTRACT

GoodCents created the daylight savings to standard time conversion code to simplify a tedious program previously attempted. Prior to the simplified version, users would have negative values for the variables hour, day, and month! The new and revised code now allows users to complete the simple time conversion process with one simple program. Using the SAS® date-time value, various date and time functions, and a series of macros, the conversion code allows users to input a SAS® dataset currently in Daylight savings time and the date of the time change. We will present the clever but simple conversion code from a user's design perspective, demonstrating the simple steps required by the user, as well as from a coder's perspective, sharing code.

KEYWORDS: macro, SAS® date-time value, TIMEPART, DATEPART, YEAR, MONTH, DAY, and HOUR functions, time conversion

INTRODUCTION

The GoodCents Consulting Services team analyzes data that varies in format. The data is most commonly recorded in 15-minute observations or in a timestamp format indicating when the air conditioner is on or off. The data for the largest project records a variety of end use (i.e. air conditioner, water heater, air handling unit, as well as televisions, microwaves, and dryers) loads in 15 or 30-minute intervals. GoodCents also reviews hourly weather data since air temperatures greatly affect air conditioning and heating usage. The clients of GoodCents determine the format and detail of certain aspects of the project so it became known after a great deal of data was collected that the data needed to be in Standard Time for the large end use load research study. This created a large problem for the weather data collected from five different weather stations and for over 100 energy usage data files in the 15-minute interval format.

CREATING THE SAS® DATE-TIME VALUE

Since all of GoodCents' data files have separate fields for year, month, day, hour and minute the original idea was to simply subtract one from the hour column. This created more issues with the first of each month and midnight. The values for month were zero and there were negative ones in the hour fields! Trying to code in for each month that a value of zero should be the month minus one created more headaches.

After taking another SAS® course at the SAS® Institute a SAS® student and GoodCents employee asked the instructor of the best method to solve the problem. It was then suggested to create a SAS® date-time variable. The SAS® date variable was created using the MDY function using of course the individual month, day, and year fields as shown below. The following statement puts the newly created "date" variable into "date9." format. Then the hour and minute fields were concatenated to the SAS® date value; this was where the hour was subtracted to achieve to standard time. The new SAS® date-time variable was then formatted to "datetime20." for ease.

```
do;  
    Date = MDY(MO,DY,YR);  
    Character_DateTime = Put(date,date9.)!!':'!! left(put(HR-1,z2.))  
    !!':'!!left(put(MN,z2.));  
    SASDateTime = Input(Character_dateTime,datetime20.);  
end;
```

EXTRACTING THE NEW VALUES TO NEW FIELDS

To keep in the same format with separate fields for year, month, day, hour, and minute it was necessary to extract that data from the SAS® date-time value. GoodCents users frequently analyze data that has separate variables for year, month, date, hour, minute, and occasionally seconds and therefore desire the new converted data sets to remain in a similar format. First, it was necessary to extract the date portion from the SAS® date-time value. This was completed using the Datepart function on the SAS® date-time variable named "SASDateTime." Then Month, Day, and Year functions were applied to the Datepart variable successfully creating a new and separate column for the month, date, and year (Karp, Andrew, Working with SAS® Date and Time Functions). Next the time portion was extracted using the Timepart function. New fields for hour and minute were created utilizing the Hour and Minute functions, for other applications the Second function would be used as well. The first two lines of code below display the use of the Datepart and Timepart functions. The Month, Day, Year, Hour, and Minute functions follow below.

```

date1=datepart(SASDateTime);
time1=timepart(SASDateTime);
NewMO=MONTH(Date1);
NewDY=DAY(Date1);
NewYR=YEAR(Date1);
NewHR=HOUR(time1);
NewMin=MINUTE(time1);
run;

```

MACRO PROCESS

The conversion code was inserted into a macro to handle multiple datasets at once. It was also helpful to include the conversion code within a macro to specify the date and time of the time change, since daylight savings time begins and ends on different dates each year. For GoodCents' specific case the date of time change varied by customer. In total there were 137 customer files resulting in 137 macro mend statements. The macro variables, as shown in the first line of code below, are sitename, site, logger, year, month, day, hour, and minute when the time conversion occurred. The sitename, site, and logger are all variables that indicate the specific customer. It was necessary to ensure the hour conversion took place at the proper hour and minute. The MACRO call statement with the macro variables is shown below followed by example MACRO mend statements. The conversion code will be ran for each MACRO mend statement.

```

%macro TimeConvert(Sitename, Site, Logger, YR, MO, DY, HR, MIN);

%mend TimeConvert;
%TimeConvert (johndoe100,101,10246,2007,11,10,06,01);
%TimeConvert (janedoe105,103,3195,2007,11,06,00,01);
%TimeConvert (petergrin110,104,3203,2007,11,07,23,01);

```

FITTING THE CONVERSION TO THE DATA

The code would not function properly if it did not have direction as to when to apply the conversion code. It was decided that the best method would be to create a new data set, which deletes all data after the date specified in the MACRO mend statement and apply the conversion code to the remaining data. This would take the specific date, hour, and minute and move the time back one hour. If the conversion occurred on November 10, 2007 at 6:10 am for example with the first observation at August 8, 2007 everything between August 8th and November 10, 2007 would be shifted back one hour. For the early morning hour on the 10th of November - 6:10 am, the macro variable time, would become 5:10 am, everything earlier that morning would be shifted back one hour as well. After the conversion is completed the code will then extract the new year, month, date, hour, and minute values. Another data step is included to delete the non-converted time before November 10, 2007 6:10 am, as in the previous example. The 6:10 am observation is kept to prevent a jump in the data from 5 to 7 am. As you may notice the record will be the same for 5:10 and 6:10 am. For GoodCents' purposes this does not create any problems and works perfectly for their needs. The two SAS® data sets are merged into one data set with the data converted to Standard time before the specified date and the data after the specified date that is already in standard time.

Data was deleted from the temporary dataset by using two IF statements utilizing the macro variables as shown below. The following statements allow only the data that needs to be converted to remain.

```

if mo=&mo and dy>&dy then delete;
if mo=&mo and dy=&dy and hr>&hr then delete;

```

Once again data was deleted from the temporary datasets by using IF statements utilizing the macro variables as shown below. The statements below allow the remainder of the data, or in other words the data that does not need to be converted, to remain.

```

if yr=2007 and mo<&mo then delete;
if mo=&mo and dy<&dy then delete;
if mo=&mo and dy=&dy and hr<&hr then delete;

```

The datasets are then simply merged together by the date to produce one dataset with all data on Standard Time. All unneeded variables are then dropped from the dataset.

```

data test5 (drop=NewYR NewMo NewDy NewHr NewMin SASDateTime Date date1 time1 SC);
merge test3 test;
by YR MO DY HR MN;
run;

```

PROBLEMS ENCOUNTERED

There was one problem that was discovered when using this method. When reviewing the SAS® data sets that were created it was noticed that everything was converted back one hour for the dates, months, and hours except for the hour near midnight. A quick glance at the SAS® date-time values showed that the hour that was midnight now had an hour value of -1:00. Extracting the date gave the previous day, for example the SAS® date-time value of 14Oct2007:-1:00 gave the extracted values of month equal to October, date equal to 13th but hour remained -1. To solve the problem, the SAS® date-time variable was scanned using the Scan function and a new temporary variable for the character form of hour and minute was created. Then a simple if hour equaled -1 then set it to hour 23 statement remedied the problem, creating the "HR" and "MN" fields in the numeric form.

```
data test6;
set test5;
CDT2=trim(scan(Character_DateTime,3,':'));
CDT1=trim(scan(Character_DateTime,2,':'));
run;
```

```
data test8;
set test6;
if CDT1='-1' and CDT2='15' then HR=23;
if CDT1='-1' and CDT2='15' then MN=15;
if CDT1='-1' and CDT2='30' then HR=23;
if CDT1='-1' and CDT2='30' then MN=30;
if CDT1='-1' and CDT2='45' then HR=23;
if CDT1='-1' and CDT2='45' then MN=45;
run;
```

The final datasets were then created and sorted by the individual date fields as shown below, then exported to replace the CSV files with the non-converted data.

```
data test7 (drop=Character_DateTime CDT1 CDT2);
set test8;
run;

proc sort data=test7;
by YR MO DY HR MN;
run;

proc export data=test7
outfile="J:\common\Load Research Study\Data Loggers\15 Minute Data\Clean
Files\&sitename..csv"
DBMS=csv replace;
run;
```

CONCLUSION

This method of converting time has eliminated the previous SAS® program that attempted to simply subtract one hour from the hour field. The key to this program was the conversion code that was introduced by John McCall from the Atlanta SAS® Institute. The remaining code was made to fit around those three lines of code where the SAS® date-time value is being created and converted to standard time. GoodCents has modified the original program for additional applications such as the hourly weather files being collected for 25 different National Weather Service stations. This program is surprisingly simple and can be modified to fit any user's application.

REFERENCES

Karp, Andrew H. Beginning Tutorials, Paper 58. "Working with SAS® Date and Time Functions."
<http://ssc.utexas.edu/docs/sashelp/sugi/24/Begtutor/p58-24.pdf>. 1 March 2008.

ACKNOWLEDGMENTS

John McCall, Instructor, Atlanta SAS® Institute
 Bill Bland, Vice President of Consulting Services, GoodCents
 Liza Thompson, Supervisor of Load Research, GoodCents

CONTACT INFORMATION

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

Audrey Ewen
GoodCents
1075 Cooper Road, Suite 100
Grayson, Georgia 30017
Work Phone: 678-836-1083
Fax: 678-344-7184
E-mail: Audrey.ewen@goodcents.com
Web: www.goodcents.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are trademarks of their respective companies.

APPENDIX I: SAS CODE

```
/******  
/* PROGRAM: HrlyTimeConvert.sas */  
/******  
/* DATE CREATED: January 2008 */  
/* BY: Ewen */  
/******  
/* DESCRIPTION: */  
/* THIS PROGRAM converts daylight saving time to standard time for a */  
/* Load Research Study */  
/******  
/* INPUTS: */  
/* Hourly2007 */  
/******  
/* EDITED BY: DATE: */  
/******  
options mprint mlogic symbolgen;  
options label nocenter ps=32 spool;  
libname loads "J:\common\Load Research Study\Data\SAS Data";  
  
proc datasets;  
delete StdTime1 Time2 Test Time3 Test5;  
quit;  
  
%macro TimeConvert(Sitename,Site,Logger,YR,MO,DY,HR,MIN);  
data StdTime1;  
set loads.&sitename;  
if yr=8 then yr=2008;  
if yr=7 then yr=2007;  
  
if yr=2007 and mo>&mo then delete;  
if mo=&mo and dy>&dy then delete;  
if mo=&mo and dy=&dy and hr>&hr then delete;  
if yr=2008 then delete;  
if yr=8 then delete;  
if yr=97 then delete;  
do;  
Date=MDY(MO,DY,YR);  
Character_DateTime=Put(date,date9.)!!':!!! left(put(HR-1,z2.))  
!!':!!!left(put(MN,z2.));  
SASDateTime=Input(Character_dateTime,datetime20.);  
end;  
  
data Time2 (drop=YR MO DY HR MN SC);  
set StdTime1;  
date1=datepart(SASDateTime);  
time1=timepart(SASDateTime);
```

```

        NewMO=MONTH(Date1);
        NewDY=DAY(Date1);
        NewYR=YEAR(Date1);
        NewHR=HOUR(time1);
        NewMin=MINUTE(time1);
run;

data test3;
set time2;
YR=NewYR;
Mo=NewMO;
DY=NewDY;
HR=NewHR;
MN=NewMin;
run;

proc sort;
by YR MO DY HR MN;
run;

data test;
set loads.&sitename;
if yr=8 then yr=2008;
if yr=7 then yr=2007;
if yr=2007 and mo<6 then delete;
if yr=2007 and mo<&mo then delete;
if mo=&mo and dy<&dy then delete;
if mo=&mo and dy=&dy and hr<&hr then delete;
run;

proc sort;
by YR MO DY HR MN;
run;

data test5 (drop=SASDateTime Date date1 time1 SC);
merge test3 test;
by YR MO DY HR MN;
run;

data test6;
set test5;
CDT2=trim(scan(Character_DateTime,3,':'));
CDT1=trim(scan(Character_DateTime,2,':'));
run;

data test8;
set test6;
if CDT1='-1' and CDT2='15' then HR=23;
if CDT1='-1' and CDT2='15' then MN=15;
if CDT1='-1' and CDT2='30' then HR=23;
if CDT1='-1' and CDT2='30' then MN=30;
if CDT1='-1' and CDT2='45' then HR=23;
if CDT1='-1' and CDT2='45' then MN=45;
run;

data test7 (drop=Character_DateTime CDT1 CDT2);
set test8;
proc sort data=test7;
by YR MO DY HR MN;
run;

```

```
proc export data=test7
outfile="J:\common\Load Research Study\Data\15 Minute Data\Clean
Files\&siteName..csv"
DBMS=csv replace;
run;

%mend TimeConvert;
%TimeConvert (johndoe100,101,10246,2007,11,10,06,01);
```