A Time Machine: Using SAS® Macros

Adel Fahmy, Symbiance Inc., Princeton, NJ

ABSTRACT

Times of events are used in many databases. When extracted into SAS® datasets, times could be in different shapes. The different methods for data entry are usually adopted for accuracy. Based on the original database design, the extracted files may contain the time in 3 separate fields for Hours, Minutes and Seconds, or in one field that contains the whole time variable. In the one-field time, the colon ":" separator may not always be present. The time fields may follow a 24-hour or 12-hour convention with another field identifying whether it is AM or PM. In either case, the field type may be Numeric or Character.

We usually end up with too many macros; each handles one type of data entry style. Modifying those macros constantly for each new project defeats the idea of generic macros. The macro presented here handles most of the commonly used time fields producing a Full Numeric Time (used in calculating duration of events) and a Full Character Time (used for accurate reporting).

KEYWORDS

Time, Hour, Minute, Second, Numeric Time, Character Time.

AUDIENCE

Database Professionals, Programmers & Statisticians with limited or advanced SAS experience.

FORMATS OF TIME

Single-Field Time: Where the time is entered as one continuous string that includes Hours, Minutes and Seconds. The most common formats are:

- hh:mm, hh:mm:ss, hh:mm:ss

Multi-Field Time: Where the time is entered as 2 or 3 separate fields that contain Hours, Minutes, and Seconds. The most common formats are:

- hh & mm, hh & mm & ss

TYPES OF TIME

Numeric Time: Where the single field that holds the full time variable or the multiple fields that hold the time parts are all numeric. The single time field, although numeric, is not necessarily a valid SAS time that could be used for calculations of a duration period. This usually occurs in the 2 and 3 separate fields case, with the lack of online validation (at data entry stage). The online validation of the separate fields inter-relationship is usually skipped to avoid slowing down the professional data entry personnel.

Character Time: Where the single field that holds the full time variable or the multiple fields that hold the time parts are all characters. Single time field is not necessarily a valid SAS time that could be used for calculations of a duration period. For character time, anything goes: company standards, boss standards, leading or no leading zeros, entering missing time parts as 00, --, **, ##, MM, etc. With the addition of data entry errors, you will need tons of IF conditions in order to cover the possible permutations of what might have been entered.

NUMERIC VERSUS CHARACTER TIME

It is important here to indicate that in statistical tabulations, the Numeric time should be listed because this is one that has been used for calculations and classification of the population. For example, assuming that an event occurs on the same day:


On the other hand, Character time should be presented “as is” in regular reports or listings. Most data listings are used by data management personnel to validate and correct database errors. It creates more confusion if the listed time is different from the entered one.

MISSING TIME PARTS

Partial Time is meaningless. If both hours and minutes are missing this will obviously result in blank time. The value 00:00 is a valid time value and cannot be used to represent unknown time. Using it will be misleading. If the Hours are missing, the time cannot be used. If Minutes or Seconds have not been reported they will be considered to be 00.
SECONDS PART OF THE TIME

Over many years of experience, I have hardly seen a project that uses the seconds and fractions of seconds. If seconds have been used, they are usually reported as 00.0 which comes from electronic capturing or formatting the time rather than data entry. An example of electronic capturing is the time that your program starts and finishes processing in the computer. Seconds and fractions of seconds are important in very specific industries, such as sports and flight control systems (e.g. 07:30:11.2). In order to avoid unnecessary complexity to the macro, seconds have been considered but not fractions of seconds.

STANDARDIZATION

Leading zeros are very important to use, particularly in 1-part time. Also, it is a good practice to use them in 2- and 3-part time. Please note that 0 and 00 is a valid value for the hour, indicating the time is after midnight and before 1:00 am.

WHERE HAVE THE LEADING ZEROS GONE ?

In Character time, the leading zeros will always be there since they are like any other alphabetic character. However, in Numeric time, although you may input your data with leading zeros, SAS stores them without (example 0730 is stored as 730). This crucial point has to be considered when you “SUBSTR’ing the time starting from the left. The macro will take this into account during the identification process of the Hour, Minute and Second. Examples:

If INFMT = hhmm:
- 3-char time 111 the time is 01:11
- 2-char time 11 the time is 11:00
- 1-char time 1 the time is 01:00

If INFMT = hhmms:
- 5-char time 11111 the time is 01:11:11
- 4-char time 1111 the time is 11:11:00
- 3-char time 111 the time is 01:11:00
- 2-char time 11 the time is 11:00:00
- 1-char time 1 the time is 01:00:00

OTHER TIME STYLES

Valid Full Numeric Time, although the macro can easily handle it, does not need to use this or any other macro. Simply output it using SAS provided Time Formats. For time variables that include separators other than the colon (:), such as hh-mm-ss, split the time into 3 character parts using the SUBSTR or SCAN function, then use the macro to handle them. It is not recommended using the TRANSLATE function to replace the dashes with blanks, unless you are sure that the hours, minutes and seconds are represented by exactly 2 digits each.

For time provided in separate 2- or 3-Numeric parts (hh mm ss) without a field that identifies whether it is AM or PM, no need to use this macro. Simply apply the SAS provided function HMS (hour, minute, second) to produce a numeric time value. Please note that this function has the following limits: hours 0 to 23, minutes 0 to 59, seconds 0 to 59.

24-HOUR CLOCK TIME

If there is no field AMPM provided that identifies whether the time occurs during Morning AM or Afternoon PM, we will assume that the input time follows a 24-hour clock system. If the field has been provided, it will be taken into consideration to transfer the time into 24-hour clock system. Allowed values are: am, a, pm, p, upper or lower case, with or without dot separator. A missing value in AMPM field is considered to be AM.

While processing 24-hour clock, some precautions have to taken into consideration.
- Beyond 24 hour, will occur in the following day.
- Example: for 5:00 pm add 12 hours, to become 17:00.
- Example: for 12:30 pm do not add 12 hours. Leave it 12:30.
- There is no 12:30 am, it is usually reported as 0:30 (after midnight, i.e. next day).
- So, basically we have only 1 up to 11 pm.
- The order of processing is crucial here and will be further explained later in the code.

TIME WITHIN A DATE

After producing the numeric time, you can use it to sort the sequence of event times within the date of the event. The interrelationship between date and time, particularly when processing the 24-hour clock issues, is beyond the scope of this paper. We will rather focus on the time issues and avoid unnecessary complexity to the macro. If the calculated time passes
the 24-hour clock, falling in the next day, the macro will output a next day flag, set to Asterisk "*". For date issues, refer to the published Date Macro paper listed under references.

PURPOSE OF PROGRAM

To transfer data entry time in different formats, numeric or character, into a valid numeric SAS time (for calculation purposes) and a character time (for reporting purposes).

- Time entered in 3 Parts HH & MM & SS (Character or Numeric)
- Time entered in 2 Parts HH & MM (Character or Numeric)
- Time entered in 1 Part HHMMSS, HHMM, HH:MM:SS, or HH:MM (Character or Numeric)
- The resulting Full Time (Character and Numeric) TMvarC and TMvarN will be in the form:
  HH:MM (default), or HHMMSS.

MACRO COMPILING

In order to compile the macro, use the following statements before the macro definition. This will create the catalog file SASMACR.SAS7BCAT (under Microsoft Windows®). In order to use the compiled macro, the same statements are used to define where the compiled macro is located. Using the compiled macro preserves the generic code and eliminates the need for using %INC to include the source code. The macro library MAC needs to be defined only ONCE in a session. A macro library cannot be redefined or cleared until the end of the SAS session.

%LET mac = %STR (c:\_research\time\macros) ;
LIBNAME mac V8 "&mac" ;
OPTIONS MSTORED SASMSTORE = mac ;

DESCRIPTION

The process is automated through one and only one generic macro. It will create two new variables, TMvarN and TMvarC.

(0) THE CALLING PROGRAM

- To call the macro, assume that a dataset HMS3C has Input Time in 3 Separate Character Fields (hh, mm & ss) and a field (ap) identifying whether the time is AM or PM. To output the time (Character and Numeric) 24-hour clock in the form hh:mm:ss, use the following macro call. [More examples including sample data are shown in Appendix 1].

  %time ( indata = hms3c , infmt  = hh mm ss , invar  = tsth tstm tsts , ampm = ap , outlbl = Test Time , outfmt = hh:mm:ss , outvar = test ) ;

  PROC PRINT NOOBS DATA = hms3c ; RUN;

(1) TIME Macro

DEFINITION:

- The macro has 7 parameters. The STORE option will compile the macro definition in a permanent catalog for later use.

  %MACRO time ( indata = , infmt  = , invar  = , ampm = , outlbl = , outfmt = hh:mm , outvar = ) ;
  / STORE DES = 'Transfer Time into Num & Char Time' ;

MACRO INPUT PARAMETERS:

- - INDATA input dataset name to be processed.
- - INFMT format of the input time including the order of hour, minute & second. Each character representing the hour, minute & second should be represented by h, m & s respectively.
  For time in 3 separate fields: hh mm ss.
For time in 2 separate fields: hh mm.
For time in 1 field: hhmmss, hh:mm:ss, hhmm, or hh:mm.

- INVAR  input time variable name(s), in the same order as the INFMT.
For time in 3 separate fields: Time-parts Variable Names
  varH varM varS in the Exact INFMT Order hh mm ss.
For time in 2 separate fields: Time-parts Variable Names
  varH varM in the Exact INFMT Order hh mm.
For time in 1 field: Time Variable Name varTM.

- AMPM  - For 12-hour clock system, identifier for Morning & Afternoon.
- Allowed values: am, a, pm, p, upper or lower case, with or without dot separator.
- Blank value means the input time follows a 24-hour clock system.

- OUTLBL output time variable label, excluding quotes.

- OUTFMT output time format for OUTVAR
  hh:mm, hh:mm:ss (Default).

- OUTVAR output time variable name, the main body of the name. Example: test, visit, exam.
  Will be auto added: a prefix "TM" and a suffix "C" or "N".
  TMoutvarN Numeric  time variable name, using outvar with prefix TM and suffix N.
  TMoutvarC Character time variable name, using outvar with prefix TM and suffix C.

**TYPE OF INPUT VARIABLE(s):**
- Get the Type of input time field(s) INVAR (Numeric or Character), so the user does not have to specify it. Save the type in a macro variable INTYP. The macro will then execute accordingly.

\[
\%\text{LOCAL intyp } ;
\]
\[\text{PROC CONTENTS NOPRINT DATA = } &\text{indata OUT = type ( KEEP = name type ) ; RUN ;}
\]
\[\text{DATA } _\text{NULL}_ ;
\]
\[\text{SET type END = eof ;}
\]
\[\text{IF INDEX ( UPCASE ( "&invar" ) , UPCASE ( COMPRESS ( name ) ) ) = 0 THEN DELETE ;}
\]
\[\text{IF type = 1 THEN typ = "N" ;}
\]
\[\text{ELSE IF type = 2 THEN typ = "C" ;}
\]
\[\text{ELSE typ = " " ;}
\]
\[\text{IF eof THEN CALL SYMPUT ( "intyp" , typ ) ;}
\]
\[\text{RUN ;}
\]

**PROCESSING INPUT DATA:**
- The valid Minimum and Maximum values are: 0 & 23 for hours, 0 & 59 for minutes and seconds. The Blank macro variable will be used to assume a zero value for missing minutes or seconds based on INFMT. Define 3 numeric variables hhn, mmn & ssn to save the numeric parts of the time and 3 character variables hhc, mmc & ssc to save the corresponding character parts. The local macro variables hh, mm & ss will store the names of the corresponding variables. Now, get the Hour, Minute & Second parts of the input time. Validate the Parts before constructing the Full Numeric Time. Assume that missing values are represented by: m = missing, u = unknown, i = illegible.

\[
\%\text{LOCAL hh mm ss ;}
\]
\[\%\text{LET bl ank } = \%\text{STR ( 0 ) ;}
\]
\[\text{DATA } &\text{indata ;}
\]
\[\text{SET } &\text{indata ;}
\]
\[\text{MISSING m u i ;}
\]
\[\text{num } = ' 0123456789: ' ;
\]

**SPLIT INPUT VARIABLES:**
- Split INVAR into Parts (hh, mm, ss) based on INFMT. For INFMT=hh mm ss, make sure that the 3 parts of INVAR are being separated with at least 2 spaces. For INFMT=hh mm, make sure that the 2 parts of INVAR are being separated with at least 1 space. For INFMT=hh:mm:ss, hh:mm, hhmmss, or hhmm, the macro will execute based on the string length. First, it will satisfy the hour, then minute, then second. To simplify the idea, all times will have a string length of 6
characters after eliminating any existing colons (:). 5- and 3-Character times will be padded with 1 left space. Assume a zero value for missing seconds in hh:mm:ss and hhmmss INFMTs, and for missing minutes in hh:mm and hhmm INFMTs.

\[
\% F %UPCASE ( \& infmt ) = %STR (HH MM SS) \&
( \%LENGTH ( \& invar ) - \%LENGTH ( %SYSFUNC ( COMPRESS ( \& invar ) ) ) ) \geq 2
\]

\%THEN \%DO;
\%LET hh = %SCAN ( \& invar , 1 ) ;
\%LET mm = %SCAN ( \& invar , 2 ) ;
\%LET ss = %SCAN ( \& invar , 3 ) ;
\%END ;
\%__________________________________________________________________________ ;

\%ELSE \%F %UPCASE ( \& infmt ) = %STR (HH MM) \&
( \%LENGTH ( \& invar ) - \%LENGTH ( %SYSFUNC ( COMPRESS ( \& invar ) ) ) ) \geq 1
\%THEN \%DO;
\%LET hh = %SCAN ( \& invar , 1 ) ;
\%LET mm = %SCAN ( \& invar , 2 ) ;
\%LET ss = %STR ( sec )        ;
\%IF %UPCASE ( \& intyp ) = N \%THEN %DO; sec = .  ; %END ;
\%ELSE %IF %UPCASE ( \& intyp ) = C %THEN %DO; sec = no ; %END ;
\%END ;
\%__________________________________________________________________________ ;

\%ELSE \%F %UPCASE ( \& infmt ) = %STR (HH:MM:SS) OR
%UPCASE ( \& infmt ) = %STR (HH:MM)
\%THEN \%DO;
\%LET hh = hh ;  \%LET mm = mm ;  \%LET ss = ss ;
\IF VERIFY ( COMPRESS ( \& invar ) , num ) = 0
\THEN DO ;
\IF 7 \leq LENGTH ( COMPRESS ( \& invar ) ) \leq 8
\THEN DO ;
hh = SCAN ( COMPRESS ( \& invar ) , 1 , `::`) ;
mn = SCAN ( COMPRESS ( \& invar ) , 2 , `::`) ;
ss = SCAN ( COMPRESS ( \& invar ) , 3 , `::`) ;
END ;
ELSE IF 4 \leq LENGTH ( COMPRESS ( \& invar ) ) \leq 6
\THEN DO ;
hh = SCAN ( COMPRESS ( \& invar ) , 1 , `::`) ;
mn = SCAN ( COMPRESS ( \& invar ) , 2 , `::`) ;
ss = ` ' ;
END ;
ELSE IF 2 \leq LENGTH ( COMPRESS ( \& invar ) ) \leq 3
\THEN DO ;
hh = SUBSTR ( COMPRESS ( \& invar ) , 1 , 2 ) ;
mn = ` ' ;
ss = ` ' ;
END ;
END ;
\]
NUMERIC TIME PARTS:
- Transfer Time Parts into Numeric if they were provided as Characters. Put Leading Zeros in front of Single Digit Hour, Minute, or Second.

    FORMAT hhn mmn ssn z2. ;

    %F %UPCASE ( &ntyp ) = N
    %THEN %DO ;
        IF &hh ne . THEN hhn = PUT ( &hh*1 , z2. ) ;
        IF &mm ne . THEN mmn = PUT ( &mm*1 , z2. ) ;
        IF &ss ne . THEN ssn = PUT ( &ss*1 , z4. ) ;
    %END ;

CHARACTER TIME PARTS:
- Construct a Character Time from input data as is. Replace Blank seconds (in hh:mm:ss) or Blank minutes (in hh:mm) with zeros (00).

    FORMAT hhc mmc ssc $2. ;

    %F %UPCASE ( &ntyp ) = N
    %THEN %DO ;
        IF    &hh NE . & VERIFY ( COMPRESS ( &hh ) , num ) NE 0
            THEN  hhc = LEFT ( PUT ( &hh , 2. ) ) ;
        ELSE  IF  &hh NE . THEN  hhc = LEFT ( PUT ( hhn , z2. ) ) ;
        /* [ The process is identical for minutes mm and seconds ss] */
    %END ;

*/ [ The process is very similar to the previous for &INFMT=HH:MM:SS and HH:MM ] */
*/ [ but use the SUBSTR function instead of SCAN function ] */
/* [ The process is identical for minutes mm and seconds ss] */

%END;
%____________________________________________________________________________;

IF hhn>= 0 & mmn = & ssn= THEN DO ; mmn=0; mmc=000; ssn=0; ssc=000; END;
ELSE IF hhn>= 0 & mmn>=0 & ssn= THEN DO ; ssn=0; ssc=000; END;

FULL CHARACTER TIME:
• Based on requested output format OUTFMT, add 1 or 2 colons as necessary. Put the Character Time Parts together in one piece.

LABEL tm&outvar.c = "&outlbl - Char" ;

% F %UPCASE (&outfmt) = %STR (HH:MM:SS) OR
  %UPCASE (&outfmt) = %STR (HMM:SS) THEN %DO; FORMAT tm&outvar.c $8.; %END;
%ELSE % F %UPCASE (&outfmt) = %STR (HH:MM) OR
  %UPCASE (&outfmt) = %STR (HMM) THEN %DO; FORMAT tm&outvar.c $5.; %END;
%ELSE %DO; FORMAT tm&outvar.c $5.; %END;

SELECT ( UPCASE ( "&outfmt" ) ) ;
  WHEN ("HH:MM:SS") tm&outvar.c =PUT(hhc,$2.)||':'||PUT(mmc,$2.)||':'||PUT(ssc,$2.);
  WHEN ("HHMMSS") tm&outvar.c =PUT(hhc,$2.)||':'||PUT(mmc,$2.)||':'||PUT(ssc,$2.);
  WHEN ("HH:MM") tm&outvar.c =PUT(hhc,$2.)||':'||PUT(mmc,$2.);
  WHEN ("HHMM") tm&outvar.c =PUT(hhc,$2.)||':'||PUT(mmc,$2.);
  OTHERWISE tm&outvar.c =PUT(hhc,$2.)||':'||PUT(mmc,$2.);
END ;

RESTORE THE ORIGINAL:
• Restore the original time variable if the constructed character time is Blank.

% F %UPCASE ( &ntyp ) = C THEN %DO ;
  IF COMPRESS ( tm&outvar.c ) = ' ' THEN tm&outvar.c = TRIM ( LEFT ( PUT ( &&invar. , $8. ) ) ) ;
%END ;
%ELSE % F %UPCASE ( &ntyp ) = N THEN %DO ;
  IF tm&outvar.c = ' ' AND &&invar. < 999999 THEN tm&outvar.c = TRIM ( LEFT ( PUT ( &&invar. , 8. ) ) ) ;
%END ;
  IF COMPRESS ( tm&outvar.c ) IN ( ': ':'::' ) THEN tm&outvar.c = ' ' ;

ADJUST FORMAT LENGTH:
• Add 1 or 2 Colons to LOUTFMT (the length of OUTFMT) then check the length of TM&outvar.C.

SELECT ( UPCASE ( COMPRESS ( "&outfmt" ) ) ) ;
  WHEN ("HH:MM:SS") loutfmt = LENGTH ( COMPRESS ( "&outfmt" ) ) ;
  WHEN ("HHMMSS") loutfmt = LENGTH ( COMPRESS ( "&outfmt" ) ) + 2 ;
  WHEN ("HH:MM") loutfmt = LENGTH ( COMPRESS ( "&outfmt" ) ) ;
  WHEN ("HHMM") loutfmt = LENGTH ( COMPRESS ( "&outfmt" ) ) + 1 ;
  OTHERWISE loutfmt = LENGTH ( COMPRESS ( "&outfmt" ) ) ;
END ;

FULL NUMERIC TIME:
• Construct a Numeric Time from Valid input data after validating the time parts.
%F %UPCASE ( &outfmt )=%STR( HH:MM SS) %THEN %DO; %LET outfmt = %STR(time8.); %END;
%ELSE %F %UPCASE ( &outfmt )=%STR( HHMMSS ) %THEN %DO; %LET outfmt = %STR(time8.); %END;
%ELSE %F %UPCASE ( &outfmt )=%STR( HH MM ) %THEN %DO; %LET outfmt = %STR(hhmm5.); %END;
%ELSE %F %UPCASE ( &outfmt )=%STR( HHMM ) %THEN %DO; %LET outfmt = %STR(hhmm5.); %END;
%ELSE %DO; %LET outfmt = %STR(hhmm5.); %END;
FORMAT tm&outvar.n &outfmt ;
LABEL tm&outvar.n = "&outlbl - Num" ;

%F %UPCASE ( &infmt ) = %STR( HH MM SS) OR %UPCASE ( &infmt ) = %STR( HH MM ) OR %UPCASE ( &infmt ) = %STR( HHMMSS ) OR %UPCASE ( &infmt ) = %STR( HH:MM ) OR %UPCASE ( &infmt ) = %STR( HHMM ) %THEN %DO;

IF 1 <= LENGTH ( COMPRESS (hhn)) <= 2 &
1 <= LENGTH ( COMPRESS (mmn)) <= 2 &
1 <= LENGTH ( COMPRESS (ssn)) <= 2 &

VERIFY ( TRIM ( LEFT (hhn)) , num ) = 0 &
VERIFY ( TRIM ( LEFT (mmn)) , num ) = 0 &
VERIFY ( TRIM ( LEFT (ssn)) , num ) = 0
THEN DO;

IF ssn >= 60 THEN DO;
  mmn = mmn + FLOOR ( ssn / 60 ) ;
  ssn = MOD ( ssn , 60 ) ;
END ;

IF mmn >= 60 THEN DO ;
  hhn = hhn + FLOOR ( mmn / 60 ) ;
%F &ampm NE %STR ( ) %THEN %DO ;
  IF &ampm NE ' ' THEN DO ;
    IF UPCASE ( SUBSTR ( COMPRESS ( TRANSLATE ( &ampm ' ,'. ' ) ) ) , 1 , 1 ) = ' P' 
    AND hhn = 12 THEN hhn = 0 ;
    %END ;
  END ;
%END ;
END ;

%F &ampm NE %STR ( ) %THEN %DO ;
  IF &ampm NE ' ' THEN DO ;
    IF (UPCASE ( SUBSTR ( COMPRESS ( TRANSLATE ( &ampm ' ,'. ' ) ) ) , 1 , 1 ) ) = ' P' 
    AND ( 1 <= hhn <= 11 ) ) OR 
    (UPCASE ( SUBSTR ( COMPRESS ( TRANSLATE ( &ampm ' ,'. ' ) ) ) , 1 , 1 ) ) = ' A' 
    AND ( hhn = 12 ) ) 
    THEN hhn = hhn + 12 ;
END ;
%END ;

IF hhn >= 24 THEN DO ;
  hhn = MOD ( hhn , 24 ) ;
nxt day = '*';
END ;
CONCLUSION

This macro will transfer time of different types and formats into full usable Numeric and Character time. The macro covers most of the commonly used data entry formats. The macro is Generic and does not need to be modified for specific projects. It handles missing information as well as am/pm issues. There is no longer a need for many different macros to handle different data entry shapes of time.

FUTURE WORK

For systems that use standard naming conventions, some of the macro input parameters could be eliminated. By using some of the SAS Functions, those parameters could be automatically identified by the macro, minimizing the user requirements. For example, if the original variables have standard naming convention, such as TESTH, TESTM & TESTS, then the OUTVAR can be calculated instead of being input:

```
%LET outvar = %SUBSTR ( &hh , 1 , %LENGTH (&hh) - 1 ) ;
```

The macro could be utilized in Edit Checks of time by comparing the resulting character and numeric times. If a complete character time was provided but results in no numeric one, then the time should be flagged with an error message.

VALIDATION

The macro and sample calls have been fully tested and validated using SAS Version 8.02 and 6.12 software on Microsoft Windows® platform.

REFERENCES

- SAS Base, Language Reference, V6, E1, Feb 1996.

AUTHOR

Adel has been a SAS user for 20 years. He worked as Independent Consultant, Sr. Statistical Programmer and Associate Director of Systems; at major pharmaceutical companies, clinical research organizations, and universities, teaching SAS to faculty and students. His special achievements include Generic Macros, Edit Checks, Database Design, Menu-Driven Systems and Optimization Techniques.

Adel has BS in Mathematics, Graduate Diploma in Systems Design, MS and M.Phil. in Computer Science from Nottingham University, UK.

CONTACT

Your comments and questions are welcomed. The author can be contacted at:

Adel Fahmy, EMS: Adel.Fahmy@netzero.net Voice: (732) 422 6651

COPYRIGHTS

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. ® indicate USA registration. Windows® is a registered trademark of Microsoft Inc. Other brand and product names are registered trademarks or trademarks of their respective companies.
APPENDIX 1

Time Macro Validation: Sample Data and Calls

- You do not have to specify type (N or C) of any of the input time fields. The macro will find out and execute accordingly.
- Missing values are represented by: m = missing, u = unknown, i = illegible.
- All test data records use a delimiter 0 to ensure reading fixed record length format.
- The columns of the test data displayed under the CARDS statement are made wider for clarity.
- Time that falls in the next day is flagged with "*".
- Test Data: following are some of the most common styles of input times.

**Example 1: HMS3C**

- **Input** 3-Char parts hh mm ss & ampm  
  **Output** hh:mm:ss

```plaintext
DATA hms3c;  
  INPUT @1 rec 2. @6 tsth $2. @11 tstm $2. @16 tsts $2. @21 ap $4. ;  
  CARDS ;  
  0   1    2    3    a      0  
  1   6    60   60   am     0  
  2   12   12   12   a.m.   0  
  3   12   59   70   p      0  
  4   0    0    0         0  
  5   5    90        p.m.   0  
  6   11        U         0  
  7                         0  
  8                         .  
  9  M    U    I         M :U :I  
; run ;

%time ( indata = hms3c ,  
  infmt = hh mm ss ,  
  invar = tsth tstm tsts ,  
  ampm = ap ,  
  outlbl = Test Time ,  
  outfmt = hh:mm:ss ,  
  outvar = test ) ;

PROC PRINT NOOBS DATA = hms3c ; RUN ;
```

**Example 2: HMS3C**

- **Input** 3-Char parts hh mm ss & ampm  
  **Output** hh:mm

```plaintext
DATA hms3c;  
  INPUT @1 rec 2. @6 tsth $2. @11 tstm $2. @16 tsts $2. @21 ap $4. ;  
  CARDS ;  
  0   1    2    3    a    01:02  
  1   6    60   60   am   06:60  
  2   12   12   12   a.m. 12:12  
  3   12   59   70   p   12:59  
  4   11   59   60   pm 11:59  
  5   0    0         00:00  
  6   5    90        p.m. 05:90  
  7   11        U         11:00  
  8                                      .  
  9  M    U    I         M :U :I  
; run ;

%time ( indata = hms3c ,  
  infmt = hh mm ss ,  
  invar = tsth tstm tsts ,  
  ampm = ap ,  
  outlbl = Test Time ,  
  outfmt = hh:mm ,  
  outvar = test ) ;

PROC PRINT NOOBS DATA = hms3c ; RUN ;
```

**Example 3: HMS2C**

- **Input** 2-Char parts hh mm & ampm  
  **Output** hh:mm

```plaintext
DATA hms3c ; SET hms3c ;  
  KEEP rec tsth tstm ap ; RUN ;

%time ( indata = hms3c ,  
  infmt = hh mm ,  
  invar = tsth tstm ,  
  ampm = ap ,  
  outlbl = Test Time ,  
  outfmt = hh:mm ,  
  outvar = test ) ;

PROC PRI NT NOOBS DATA = hms3c ; RUN ;
```

```plaintext
rec tsth tstm ap tmtestc tntestn  
0 1 2 3 a 01:02 00:00 01:02  
1 6 60 60 am 06:60 00:00 07:00  
2 12 12 12 a.m. 12:12 00:00 12:12  
3 12 59 70 p 12:59 00:00 12:59  
4 11 59 60 pm 11:59 00:00 23:59  
5 0 0 0 00:00 00:00 00:00  
6 5 90 p.m. 05:90 00:00 18:30  
7 11 U 11:00 00:00 11:00  
8                                      .  
9  M U I M:U M:U I  
```
Example 2: HMS3N

**Input** 3-Num parts hh mm ss & ampm
**Output** hh:mm:ss

```sas
DATA hms3n;
    INPUT @1 rec 2. @6 tsth 2. @11 tstm 2. @16 tsts 2. @21 ap $4. ;
    CARDS;
    0 1 2 01:02:03 1:02:03
    1 6 60 06:60:60 7:00:00
    2 12 12 12:12:12 0:12:12 *
    3 12 59 12:59:70 13:00:10
    4 11 59 11:59:60 0:00:00
    5 0 0 00:00:00 0:00:00
    6 5 90 05:90:00 18:30:00
    7 11 . U 11:00:00 11:00:00
    8 . .
    9 M U I M :U :I
    ;
    %time ( indata = hms3n ,
    infmt = hh mm ss ,
    invar = tsth tstm tsts ,
    ampm = ap ,
    outlbl = Test Time ,
    outfmt = hh:mm:ss ,
    outvar = test ) ;
PROC PRI NT NOOBS DATA = hms3n ; RUN ;
```
```sas
rec tsth tstm tsts ap tmtestc tmtestn
0 1 2 01:02:03 1:02:03
1 6 60 06:60:60 7:00:00
2 12 12 12:12:12 0:12:12 *
3 12 59 12:59:70 13:00:10
4 11 59 11:59:60 0:00:00
5 0 0 00:00:00 0:00:00
6 5 90 05:90:00 18:30:00
7 11 . U 11:00:00 11:00:00
8 . .
9 M U I M :U :I
```

**Input** 3-Num parts hh mm ss & ampm
**Output** hh:mm

```sas
%time ( indata = hms3n ,
infmt = hh mm ss ,
invar = tsth tstm tsts ,
ampm = ap ,
outlbl = Test Time ,
outfmt = hh:mm ,
outvar = test ) ;
PROC PRI NT NOOBS DATA = hms3n ; RUN ;
```
**Input** 2-Num parts hh mm & ampm  
**Output** hh:mm

``` SAS 
DATA hms3n; SET hms3n;  
KEEP rec tsth tstm ap; RUN ;
%time ( indata = hms3n ,  
infmt = hh mm ,  
invar = tsth tstm ,  
ampm = ap ,  
outlbl = Test Time ,  
outfmt = hh:mm ,  
outvar = test ) ;
PROC PRINT NOOBS DATA = hms3n ; RUN ;
```

**Example 3: HMS1C_**

**Input** 1-Char part hh:mm:ss & ampm  
**Output** hh:mm:ss

``` SAS 
DATA hms1c_;  
INPUT @1 rec 2. @6 tsttm $8. @17 ap $4. ;
CARDS ;
0 01:02:03 a 0  
1 06:60:60 am 0  
2 12:12:12 a.m 0  
3 12:59:70 p 0  
4 11:59:60 pm 0  
5 00:00:00 0  
6 05:90: p.m 0  
7 11::00  
```

``` SAS 
rec tsth tstm ap tntestc tntestn  
0 1 2 3 a 01:02 1:02  
1 6 60 60 am 06:60 7:01  
2 12 12 12 a.m 12:12 0:12 *  
3 12 59 70 p 12:59 13:00  
4 11 59 60 pm 11:59 0:00  
5 0 0 0 00:00 0:00  
6 5 90 p.m 05:90 18:30  
7 11 . U 11:00 11:00  
8 . .  
9 M U I M :U .
```

``` SAS 
rec tsth tstm ap tntestc tntestn  
0 1 2 3 a 01:02 1:02  
1 6 60 60 am 06:60 7:01  
2 12 12 12 a.m 12:12 0:12 *  
3 12 59 70 p 12:59 13:00  
4 11 59 60 pm 11:59 0:00  
5 0 0 0 00:00 0:00  
6 5 90 p.m 05:90 18:30  
7 11 . U 11:00 11:00  
8 . .  
9 M U I M :U .
```

``` SAS 
DATA hms3n; SET hms3n;  
KEEP rec tsth tstm; RUN ;
%time ( indata = hms3n ,  
infmt = hh mm ,  
invar = tsth tstm ,  
ampm = ,  
outlbl = Test Time ,  
outfmt = hh:mm:ss ,  
outvar = test ) ;
PROC PRINT NOOBS DATA = hms3n ; RUN ;
```
Example 4: HMS1C

- Input: 1-Char part hhmmss & ampm
- Output: hh:mm:ss

```plaintext
DATA hms1c;
  INPUT @1 rec 2. @6 tsttm $6. @17 ap $4.;
  CARDS;
  0 10203 a 01:02:03 1:02:03
  1 66060 am 06:60:60 7:01:00
  2 121212 a.m. 12:12:12 0:12:12 *
  3 125970 p 12:59:70 13:00:10
  4 115960 pm 11:59:60 0:00:00
  5 0 00:00:00 0:00:00
  6 590 p.m. 05:90:00 18:30:00
  7 11 11:00:00 11:00:00
  8
  9 UNK UNK
; RUN;
```

Example 5: HMS1N

- Input: 1-Num part hhmmss & ampm
- Output: hh:mm:ss

```plaintext
DATA hms1n;
  INPUT @1 rec 2. @6 tsttm 6. @17 ap $4.;
  CARDS;
  0 10203 a 0
  1 66060 am 0
  2 121212 a.m. 0
  3 125970 p 0
  4 115960 pm 0
  5 0 0
  6 590 p.m. 0
  7 11 0
  8
  9 U 0
; RUN;
```

Example 5: HMS1N

```plaintext
DATA hms1n;
  INPUT @1 rec 2. @6 tsttm 6. @17 ap $4.;
  CARDS;
  0 10203 a 0
  1 66060 am 0
  2 121212 a.m. 0
  3 125970 p 0
  4 115960 pm 0
  5 0 0
  6 590 p.m. 0
  7 11 0
  8
  9 U 0
; RUN;
```