Standard SAS Macros for Standard Date/Time Processing

Qin Lin, Applied Clinical Intelligence, LLC, Bala Cynwyd, PA
Tim Kelly, Applied Clinical Intelligence, LLC, Bala Cynwyd, PA

ABSTRACT:

Processing partial or invalid date/time values to meet certain needs imposes challenges for all SAS programmers. The CDISC Study Data Tabulation Model (SDTM) standard has been added as a draft FDA guidance. It specifies the standard for presenting full/partial date/time values. To follow the standards of CDISC SDTM v3.1, we developed several dynamic macros to process dates and times.

1. %CDISCDTM: reads in most date/time values (complete, partial or invalid), checks its validity and converts date/time to standard ISO8601 format;
2. %YEAR4: converts two-digits year to four-digits year;
3. %DURATION: calculates the duration between two dates;
4. %STUDYDAY: calculates the study day;
5. %TEAE: defines treatment emergent adverse events based on AE start date and reference date, as well as lag time;
6. %MINLEVEL: identifies the lowest common date part that exists between two dates; it is called by %DURATION, %STUDYDAY, %TEAE.

These macros provide a useful tool for modifying date/time variables to the required standard and also assist in the creation of other required standard variables.

INTRODUCTION:

Depending on the source data, date/time values may contain complete, partial, or invalid dates. When dealing with such data, many issues may be encountered. Usually manipulation on those date/time values must occur before they can be used in computation. Since more clients are demanding the datasets follow the CDISC SDTM compliant format, we developed a series of standard macros to meet the CDISC SDTM implementation guide v3.1 requirement and process dates/times in a more efficient way.

All macros introduced in this paper are related to date/time conversion or computation. First, we developed a fundamental macro called %CDISCDTM to convert all the dates/times into ISO 8601 format. This macro is very powerful in processing partial, invalid and missing date values. The rest of the macros were then developed to allow for dynamic processing when creating as well as using CDISC compliant data sets.

STANDARD MACROS:

I: %CDISCDTM (ordate, ortime, outvar=):

1. **Introduction:**
   This fundamental macro is developed to convert date/time variables into ISO 8601 format. According to CDISC SDTM Implementation Guide (SDS Version 3.1), the template for ISO 8601 sets dates as YYYY-MM-DD**T**hh:mm:ss; where:
   YYYY = four-digit year
   MM = two-digit representation of the month (01=January, etc.)
   DD = two-digit day of month (01 through 31)
   T = separator to indicate time information follows
   hh = two digits of hour (00 through 23) (am/pm is NOT allowed)
   mm = two digits of minute (00 through 59)
   ss = two digits of second (00 through 59)

For example, ISO 8601 representation of full and partial date/time values is as the following:

December 15, 2005 14:15:17	complete date	2005-12-15T14:15:17
December 15, 2005 14:15	unknown seconds	2005-12-15T14:15
December 15, 2005 14	unknown minutes	2005-12-15T14
December 15, 2005	unknown time	2005-12-15
December, 2005	unknown day	2005-12
2005	unknown month	2005

2. Parameters:

1) Ordate: the original character date variable in MMDDYY or MMDDYYYY format;
2) Ortime: the original character time variable in TIME5 or TIME8 format;
3) Outvar: the derived character date/time output variable in ISO 8601 format;
Note: All the parameters must be character variables.

3. Process date/time in the macro (see source code in Appendix):

1) It is possible that any date value can be passed into the macro, such as unknown or partial month, day or year, invalid month, day or year. In order to properly convert those date/time variables, it first extracts the character date/time value in each piece (year, month, day, hour, minute, second).

2) Then, it checks the validity level of the original date/time. If any invalid, partial or missing date/time was input, then put warning message in the log and convert the invalid part to "UU" or "UUUU" as standard unknown notation in converting those partial date/times. Depends on the value of each piece, it is handled very differently. For instance, processing the YEAR piece, it checks whether date is in mm/dd/yy format (if length(var)=8); if so, use %YEAR4 to convert the two digit year to four digit year in mm/dd/yyyy format. But the last step is similar, either it can be converted in numeric or it will be represented as unknown.

3) Final step is to reassemble each date/time piece together and put it into ISO 8601 format.

4. Examples:

The following two tables listed some examples of date/time in various format to be converted into CDISC ISO 8601 format.

Table 1: Sample Date Conversion

Input Date	Sample Date	Macro Converted Date
Good full dates	06/15/2006, 06/15/06	2006-06-15
Length of dates not equal 8 or 10 (including partial dates)	2/3/05, 02/3/05, 02/3/2005 /03/05 02//05 / /05, //05, / /2005	2005-02-03 2005 2005-02 2005
Unknown notation not using UU	06/TT/06, 06/00/2006 TT/06/06, TT/TT/06, 00/02/06 06/05/TTTT	2006-06 2006
Invalid day or month	06/31/05, 06/31/2005 19/05/06, 19/05/2006	2005-06 2006
If year is before year cut off, put warning message in the log, but still convert it.	06/15/1901	1901-06-15

Note: Input date in mmddyy8 or mmddyy10 format.

Table 2: Sample Time Conversion

Input Time	Sample Time	Macro Converted Time
Good full times	02:03:04 02:03	2006-06-15T02:03:04 2006-06-15T02:03
Length of dates not equal 5 or 8 (including partial times)	2:04:05, 2:4:05, 2:4:5 02:4, 2:4, 2:04, 02:04: 02: 02: :05 :04:05	2006-06-15T02:04:05 2006-06-15T02:04 2006-06-15T02 2006-06-15T02 2006-06-15
Unknown notation not using UU	02:TT, 02:TT:TT TT:03:04 02:03:TT	2006-06-15T02 2006-06-15 2006-06-15T02:03
Invalid hour, min or second	02:61 02:03:62 25:03:04	2006-06-15T02 2006-06-15T02:03 2006-06-15

Note: Input time in time5 or time8 format and date set as '06/15/2006'.

5. Sample Output:

```
** convert good full date/time into ISO 8601 format **;
data test;
  length testdate $10 testtime $8;

  testdate = '06/15/2005';
  testtime = '02:40:19';
  %cdiscdtm(testdate, testtime, outvar=stdtc);
run;

proc print;
  var testdate testtime stdtc;
run;
```

Output:

Obs	TESTDATE	TESTTIME	STDTC
1	06/15/2005	02:40:19	2005-06-15T02:40:19

II: %YEAR4(yvar, outvar=, newunk=UUUU);

1. Introduction:

Since the ISO 8601 format needs the output date in YYYY format and it is common that dates are in mm/dd/yy format, we created the %YEAR4 macro to convert those two digit years into four digit years to meet this requirement. In the %CDISCDTM macro, when the original date was in mm/dd/yy format, the %YEAR4 macro was called to convert the two digit year into four digit year.

2. Parameters:

yvar = two digit year which is a character variable to be converted to four digit character year;
outvar = the output year variable in four character digits;
newunk = the user defined value for the unknown two digit year; here we use four characters to represent the unknown year, default value=UUUU since %CDISCDTM uses UUUU as the unknown year representation;

3. Details of SAS Code (see source code in appendix):

This macro begins with checking whether the year variable is valid for conversion. If unknown or invalid two digit year, for instance: '0a', '1u', 'TT', does not use 'UU' as the representation for the unknown year, it will be converted to 'UUUU' to be accepted in the %CDISCDTM macro. If two digit year is valid, it compares the year value with the year cut off option value to decide which century should be used for

conversion into four digit year. This macro also handles special case like '00', which will be hard coded as year='2000'.

4. **Sample Output:**

```
** pass in two digit year to convert into four digit year **;
data test;
  length testyr $2 ctestyr $4;

  testyr='15';
  %year4(testyr, outvar=ctestyr);
run;

proc print;
run;
```

Output:

Obs	TESTYR	CTESTYR	TEMPVAR	CENT
1	**15**	**1915**	**15**	**1900**

III: %MINLEVEL:

1. **Introduction:**
   This macro is used to identify the common date/time parts that exist between the two date/time variables. This macro is called by the three macros (%DURATION, %STUDYDAY, %TEAE). Following the ISO 8601 format, all the passed in date/time variables can be defined with the following levels:

   level 1 = yyyy
   level 2 = yyyy – mm
   level 3 = yyyy – mm – dd
   level 4 = yyyy – mm – ddThh
   level 5 = yyyy – mm – ddThh:mm
   level 6 = yyyy – mm – ddThh:mm:ss

   These six level definitions are essential for our macro development. For instance, in order to calculate the duration between the two dates, it has to identify each part of the date/time variables, how many years, how many months, how many days etc. By identifying those pieces of the date/time variable, it makes the other macros (%DURATION, %STUDYDAY, %TEAE) easier for implementation.

2. **Parameters:**
   No parameters are necessary since all the macro variables used in this macro are passed in by the macro calling it.

3. **Function of %MINLEVEL (see source code in appendix):**
   1) Find out the minimum level of the start or end date/time variable.
      Since the date/time variables are already in ISO 8601 format, it is easy to identify the level of the date/time variable by scanning for '-' or ':' to get each piece (year, month, day, hour, minute, second) and count the level.
   2) Find the highest level of data in common for both start date/time and end date/time to define the minimum level of the date/time.
   3) Convert each character piece into numeric for calculations in other macros.
   4) It will be called by %DURATION, %STUDYDAY and %TEAE.

IV: %DURATION (sdate, edate, outcome):

1. **Introduction:**
   Duration, such as AE duration or treatment duration, is frequently used in the listings section of a clinical study report. However, only when both start date/time (--STDTC) and end date/time (-- ENDTC) exist, can we calculate the duration as the difference of these two values; otherwise, the duration is considered as missing. It is easy to implement this macro by calling the previously introduced %MINLEVEL macro.

2. **Format for the duration result: nYnMnDTnHnMnS or nW;**
   According to CDISC SDTM Implementation Guide (SDS Version 3.1), the format for the duration result will be nYnMnDTnHnMnS or nW, where the 'n' preceding each letter represents the number of Years, Months, Days, Hours, Minutes, Seconds, or Weeks; 'T' is used to separate the date and time components in the ISO 8601 date/time format.

3. **Parameters:**
   1) sdate: the start date (i.e. the start date of AE or CM, or study reference date); character date in ISO 8601 format;
   2) edate: the end date (i.e. the end date of AE or CM); character date in ISO 8601 format;
   3) outcome: the variable to hold the duration result; for CDISC compliance, the result is in nYnMnDTnHnMnS or nW format;

4. **Calculate the duration in the macro:**
   1) Call %MINLEVEL to determine the minimum level of the date/time variables;
   2) Calculate each piece of the duration based on the difference between start and end date/time variables;
   3) Get each piece of the duration (Y, M, D, H, M, S or W) and display in the required format;

5. **Sample Output:**

```
** Calculate AE duration **;
data test;
  length aestdtc aeendtc $20 aedur $50;

  aestdtc = '2005-11-15T14:15:17';
  aeendtc = '2005-12-16T08:12:15';

  %duration(aestdtc, aeendtc, aedur);

  keep aestdtc aeendtc aedur;
run;

proc print;
run;
```

Output:

Obs	AESTDTC	AEENDTC	AEDUR
1	**2005-11-15T14:15:17**	**2005-12-16T08:12:15**	**04w02dT17h56m58s**

V: %STUDYDAY (sdate, edate, outcome):

1. **Introduction:**
   Usually there's a study day variable (--DY) in CDISC compliance datasets, which is considered the descriptive representation of a relative date within the study. There's also a reference date which identifies the start date of the study, the RFSTDTC variable in Demography domain, which is identified as Day 1. Day 0 is not allowed in CDISC study day calculation. This macro was developed to calculate

the relative study day between the date portion of any DTC variable (in ISO 8601 format) to the date portion of the study reference date.

2. Parameters:

1) sdate: the start date (i.e. study reference date: RFSTDTC); character date in ISO 8601 format;
2) edate: the end date (i.e. the lab date); character date in ISO 8601 format;
3) outcome: the variable to hold the study day result; in days;

3. Calculate the study day in the macro:

1) Call %MINLEVEL to determine the minimum level of the date/time variables;
2) The rule for calculating the study day:
   a) if –DTC is on or after RFSTDTC, then study day = (date part of –DTC) – (date part of RFSTDTC) +1;
   b) if –DTC is prior to RFSTDTC, then study day = (date part of –DTC) – (date part of RFSTDC);
   Note: all of the study day values are integers; the result is in days.

4. Sample Output:

```
** Calcuate Study Day for Vitals Dataset **;
data test;
  length vtdtc rfstdtc $20;

  vtdtc='2005-11-16';
  rfstdtc='2005-11-11';

  %studyday(rfstdtc, vtdtc, vtdy);

  keep vtdtc rfstdtc vtdy;
run;

proc print;
run;
```

output:

Obs	VTDTC	RFSTDTC	VTDY
1	**2005-11-16**	**2005-11-11**	**6**

VI: %TEAE (outcome, datevar, srefdate, erefdate=, lag=0):

1. Introduction:

Usually treatment emergent adverse events (TEAE) are commonly reported in clinical study reports. To determine whether the adverse event is treatment emergent, we need to identify whether the events occurred within the sponsor specified range. To be conservative in reporting TEAE, when the AEs are proven to occur outside of the specified window, we consider it as a non-treatment emergent event, TEAE=No; otherwise, it is TEAE. In most cases, the sponsor will specify the range as start of study medication date to the end of study drug plus lag time (e.g. 30 days). We may need to compare the AE start date with the study medication start date and end of study date plus some days to determine whether the event is a TEAE.

2. Parameter:

1) outcome: the variable to hold the outcome of the TEAE definition; possible value= 'N' or NULL, NULL means the event is a TEAE;
2) datevar: the AE event start date; character date in ISO 8601 format;
3) srefdate: the start reference date (i.e. study reference date, RFSTDTC); character date in ISO 8601 format;

4) erefdate: the end reference date; this parameter is optional, depends on the study; if it exists, it should be character date in ISO 8601 format;
5) lag: the number of days in the lag period; default set as 0 in case of no lag; must be numeric days.

3. Details of the macro:

1) Compare the AE start date with reference start date by calling %MINLEVEL macro to break down each piece of the date/time variables, then comparing each available piece to determine whether the AE start date/time is out of the specified date/time window and set the flag as 'N' (not a TEAE);

2) If end reference date exists, compare the AE start date with the end reference date by calling %MINLEVEL macro as well. If lag time exists, we will compare the dates from level 3 which means both dates have the complete date part; otherwise, if the end reference date is a full date, we will still compare after adding the lag time to the partial start date; if both the AE start date and end reference date are partial, we can't add the lag time (in days) to end reference date to compare it, unless the lag time=0 (as default), we will still compare dates in level 1 or 2 to determine whether it is TEAE;

3) If the TEAE flag was not set as 'N', leave the flag as missing, so the event is considered a TEAE.

4. Sample Output:

```
** Define Treatment Emergent AEs **;
data test;
  length aestdtc rfstdtc rfendtc $20 teae $1;

  aestdtc = '2005-11-20';
  rfstdtc = '2005-11-25';
  rfendtc = '2006-01-20';

  %teae(teae, aestdtc, rfstdtc, erefdate=rfendtc, lag=30);

  keep teae aestdtc rfstdtc rfendtc;
run;
```

Output:

Obs	AESTDTC	RFSTDTC	STOPDTC	TEAE
1	**2005-11-20**	**2005-11-25**	**2006-01-20**	**N**

CONCLUSION

These macros are very powerful in processing the date/time into standard format and assist in the creation of other required standard variables. It makes programmers life easier when creating CDISC compliant data sets.

ACKNOWLEDGEMENTS

We would like to acknowledge the programming team at Applied Clinical Intelligence for testing these macros, especially thanks to Katie Burtis for her effort in developing some of the draft code.

REFERENCES

CDISC SDTM Implementation Guide (SDS Version 3.1)

CONTACT INFORMATION

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

Qin Lin
Applied Clinical Intelligence, LLC
225 City Avenue, Suite 15
Bala Cynwyd, PA 19004
Tel: 484-429-7200 ext. 16
Email: qlin@a-ci.com
www.a-ci.comT

Tim Kelly
Applied Clinical Intelligence, LLC
225 City Avenue, Suite 15
Bala Cynwyd, PA 19004
Tel: 484-429-7200 ext. 13
Email: tkelly@a-ci.com
www.a-ci.com

SOURCE CODE:

%CDISCDTM MACRO:

```
%macro cdiscdtm(ordate, ortime, outvar=);

    /* set length */
    length oryr yr $4 mon dy $2 vdate $10;

    /*************************************************************************/
    /************************** process date part ****************************/
    /*************************************************************************/

    /* first, get individual part of the date */
    if &ordate ne '' then do;

      /* warning if date not in standard format */
      if (length(&ordate) not in (8, 10)) then put
            'WARN' 'ING:(ACI) Date is parital, not using MM/DD/YYYY or MM/DD/YY controlled terminology: '
           &ordate=;

      /*********** get year part ***********/
      /* If month part missing (eg: /02/06) or day part missing (eg:01//06) */
      if substr(&ordate, 1, 1)='/' or index(&ordate, '//') ne 0 then
        oryr=substr(&ordate,(length(&ordate)-1));
      /* complete year */
      else oryr=scan(&ordate, 3, '/');

      /* two digit year: need to be converted in four digit year using %year4 macro */
      if length(oryr)=2 then do;
        %year4(oryr, outvar=yr);
        &ordate=substr(&ordate,1,(length(&ordate)-2)) || trim(left(yr));
      end;
      /* four digit year */
      else if length(oryr)=4 then do;
              yr=oryr;
             /* if four digit year can't be converted in numeric (means it contains character),
                 convert to 'UUUU' */
             if yr ne '' and input(yr, ??4.) eq . then yr='UUUU';
      end;
      /* set partial year=unknown */
      else if length(oryr) not in (2,4) then yr='UUUU';

      /* convert to numeric year */
      if yr ne '' and input(yr, ??4.) ne . then yrn=input(yr, 4.);
      /* warning if year missing or before year cutoff */
      if (yrn lt &cutoff) or (yrn eq .)  then put
        'WARN' 'ING:(ACI) Year is before year cutoff option or year is unknown, Year MUST be present: '
          &ordate= yr=;

      /*********** get month part ***********/
      /* convert to unknown for special case: if no space before missing month part:
         ordate="/02/06" */
      if substr(&ordate, 1,1)='/' then mon='UU';
      /* otherwise first piece is the month part */
      else mon=scan(&ordate,1,'/');

      /* if only one digit month: '6/3/2006', convert to two digit */
      if length(mon)=1 then mon=put(input(mon,best.), z2.);

      /* invalid or missing month, convert to unknown */
      if (mon not in('UU','01','02','03','04','05','06','07','08','09','10','11','12')) then do;
        put 'WARN' 'ING:(ACI) Month is not valid or Unknown month is not UU: ' mon=;
        /* if invalid or missing, convert to UU */
        mon='UU';
      end;

      /*********** get day part ***********/
      /* convert to unknown for special case: if no space between month/year for missing day part:
          date="02//06" */
      if index(&ordate, '//') ne 0 then dy='UU';
      /* otherwise, second piece is the day part */
      else dy=scan(&ordate,2,'/');
      /*if only one digit day: '6/3/2006', convert to two digit*/
```

```
  if length(dy)=1 then dy=put(input(dy,best.), z2.);

  /* convert to numeric */
  if dy ne '' and input(dy, ??2.) ne . then dyn = input(dy, 2.);

  /* check invalid day within the month */
  if (dyn ne .) and (yrn ne .) then do;
    if (mon in('01','03','05','07','08','10','12') and not(1 le dyn le 31)) or
       (mon in('04','06','09','11') and not(1 le dyn le 30)) or
       ((mon eq '02') and (mod(yrn,4) ne 0) and not(1 le dyn le 28)) or
      ((mon eq '02') and (mod(yrn,4) eq 0) and not(1 le dyn le 29)) then do;
         put 'ERR' 'OR:(ACI) Day is not valid: ' dy= mon= yr=;
        /* if day invalid, but final date still can be converted to yyyy-mm */
        dy='UU';
    end;
  end;

  /* convert missing or other unknown day to UU */
  if (dy ne 'UU') and (dyn eq .) then do;
      put 'ERR' 'OR:(ACI) Unknown day is not UU: ' dy= mon= yr=;
     /* if other unknown character: like 'TT' or day missing, convert to UU */
     dy='UU';
  end;

  /********** Put together date parts into full date variable **********/
  /* handled month unknown, but day known case */
  if yr not in ('', 'UUUU') then do;
    if (dy ne 'UU') and (mon ne 'UU') then vdate = yr || '-' || mon || '-' || dy;
    else if (dy eq 'UU') and (mon ne 'UU') then vdate = yr || '-' || mon;
    else if (mon eq 'UU') then vdate = yr;
    else vdate = '';
  end;
  else vdate = '';

end;
else vdate = '';


/*************************************************************************/
/************************** process time part **************************/
/*************************************************************************/
/* if time variable exists, get time part using either TIME5 or TIME8 structure */
%if &ortime ne %then %do;
    length hr min sec $2 vtime $8;

    /* First time is for hours */
    hr=scan(&ortime,1,':');
     /* if one digit, converted to be two digit */
    if length(compress(hr))=1 and compress(hr) in ('1','2','3','4','5','6','7','8','9','0') then
       hr = put(input(hr,8.), z2.);
    if (hr not in('UU','01','02','03','04','05','06','07','08','09','10','11','12','13',
                  '14','15','16','17','18','19','20','21','22','23','00')) then do;
           put 'ERR' 'OR:(ACI) Hour is not valid or Unknown hour is not UU: ' &ortime= hr=;
                  /* reset unknown or invalid to 'UU', so it will be converted correctly */
                  hr='UU';
    end;

    /* Second time is for minutes */
    min=scan(&ortime,2,':');
     /* if one digit, converted to be two digit */
    if length(compress(min))=1 and compress(min) in ('1','2','3','4','5','6','7','8','9','0') then
              min =put(input(min,8.), z2.);
     if (min not in('UU','01','02','03','04','05','06','07','08','09','10',
                    '11','12','13','14','15','16','17','18','19','20',
                    '21','22','23','24','25','26','27','28','29','30',
                    '31','32','33','34','35','36','37','38','39','40',
                    '41','42','43','44','45','46','47','48','49','50',
                    '51','52','53','54','55','56','57','58','59','00')) then do;
          put 'ERR' 'OR:(ACI) Minute is not valid or Unknown minute is not UU: ' hr= min=;
          /* reset unknown or invalid to 'UU', so it will be converted correctly */
           min='UU';
    end;

    /* Third time is for seconds if exists */
     if min not in ('','UU') then do;
```

```
          if index(substr(&ortime, index(&ortime, ':')+1), ':') ne 0 then do;
             sec=scan(&ortime,3,':');
             /* if one digit, converted to be two digit */
             if length(compress(sec))=1 and compress(sec) in ('1','2','3','4','5','6','7','8','9','0')
                   then sec =put(input(sec,8.), z2.);
              if (sec not in('UU','01','02','03','04','05','06','07','08','09','10',
                        '11','12','13','14','15','16','17','18','19','20',
                        '21','22','23','24','25','26','27','28','29','30',
                        '31','32','33','34','35','36','37','38','39','40',
                        '41','42','43','44','45','46','47','48','49','50',
                        '51','52','53','54','55','56','57','58','59','00')) then do;
                     put 'ERR' 'OR:(ACI) Second is not valid or Unknown second is not UU: ' hr= min= sec=;
                      /* reset unknown or invalid to 'UU', so it will be converted correctly */
                      sec='UU';
               end;
             end;
           end;

          /** Set time variable accordingly **/
          if (hr not in ('UU', '')) then do;
            if (sec not in ('', 'UU')) and (min ne 'UU') then vtime = hr || ':' || min || ':' || sec;
            else if (sec in ('', 'UU')) and (min ne 'UU') then vtime = hr || ':' || min;
            else if (min eq 'UU') then vtime = hr;
             else vtime = '';
          end;
          else do;
                  vtime='';
          end;

        %end;  /* End processing on time variable */
        %else %do;
                 vtime = '';
        %end;

        /*****************************************************************************/
       /************ Put together the CDISC date and time variables **************/
        /*****************************************************************************/
        if (vtime ne ' ') then &outvar = trim(left(vdate)) || 'T' || trim(left(vtime));
       else if (vdate ne ' ') then &outvar = trim(left(vdate));

%mend;
```

%YEAR4 MACRO:

```
%macro year4(yvar, outvar=, newunk=UUUU);
   /** Note: Year MUST be 2 digits and character **/
   /* if unknown code is not ‘UU’, for instance: '0a', '1x', 'TT' etc., converted to ‘UUUU’;
      ‘UU’ will be converted to ‘UUUU’ as well*/
   if indexc(upcase(&yvar),
            'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W',
             ’X','Y','Z') ne 0
     then &outvar="&newunk";
   else do;
          /* year is missing */
          if (&yvar eq '') then &outvar = '';
           else do;
                 /* hard code for special case: year=2000 */
                 if &yvar = '00' then &outvar='2000';
                 else if (input(&yvar,2.) ne .) then do;
                        /* actual year can be converted to numeric */
                         tempvar=input(&yvar,2.);
                         /* if 2 digit Year is prior to YearCutOff then century is 2000, else if it’s
                           after then century is 1900 */
                         if (tempvar lt input(substr("&yco",3,2),2.)) then cent=2000;
                         else if (tempvar ge input(substr("&yco",3,2),2.)) then cent=1900;

                         &outvar = put(cent+tempvar,4.);
                 end;
           end;
   end;
%mend;
```

%MINLEVEL MACRO:

```
%macro minlevel;

    /** Step 1: Determine level of date for start and end dates **/
    %do i = 1 %to 2;

       length dates1 dates2 $20;
       dates1 = &sdate;
       dates2 = &edate;

       timepart&i=0;
       if (index(dates&i,'T') gt 0) then timepart&i=1;
       do while (index(dates&i,':') ne 0);
         dates&i = substr(dates&i,index(dates&i,':')+1);
         timepart&i+1;
       end;

       /* Reset for Date checking */
       dates1 = &sdate;
       dates2 = &edate;

       datepart&i=0;
       if (dates&i ne '') then datepart&i=1;
       do while (index(dates&i,'-') ne 0);
         dates&i = substr(dates&i,index(dates&i,'-')+1);
         datepart&i+1;
       end;

        /** Set level of data for start and end dates
        level 1 = yyyy
        level 2 = yyyy-mm
        level 3 = yyyy-mm-dd
        level 4 = yyyy-mm-ddThh
        level 5 = yyyy-mm-ddThh:mm
        level 6 = yyyy-mm-ddThh:mm:ss   **/

       level&i=0;

            if (timepart&i eq 3) then level&i = 6;
       else if (timepart&i eq 2) then level&i = 5;
       else if (timepart&i eq 1) then level&i = 4;
       else if (datepart&i eq 3) then level&i = 3;
       else if (datepart&i eq 2) then level&i = 2;
       else if (datepart&i eq 1) then level&i = 1;

       dates1=&sdate;
       dates2=&edate;

       if (level&i ^in(1,2,3,4,5,6)) and (dates&i ne '') then
          put 'WARN' 'ING:(ACI) Date not in correct CDISC SDTM format: ' &sdate=;
    %end;


    /* Step 2: Find highest level in common */
    minlevel = min(level1,level2);
    if (level1 = . or level2 = .) then minlevel = .;



    /* Step 3: Calculate each date/time piece */
    if minlevel ge 1 then do;                       *years (only for years or years/months);
        syr = input(substr(&sdate,1,4),4.);
        eyr = input(substr(&edate,1,4),4.);
        years = eyr-syr;
    end;

    if minlevel ge 2 then do;                       *months (only if minlevel = 2, otherwise weeks);
        smon = input(substr(&sdate,6,2),3.);
        emon = input(substr(&edate,6,2),3.);
        months = emon-smon;
        if months lt 0 then do;
            years = years - 1;
            months = months + 12;
```

```
        end;
    end;

    if minlevel ge 3 then do;                       *days (for minlevels 3+);
        sday = input(substr(&sdate,1,10),yymmdd10.);
        eday = input(substr(&edate,1,10),yymmdd10.);
    end;

    if minlevel ge 4 then do;                       *hours;
        shr = input(substr(&sdate,12,2),3.);
        ehr = input(substr(&edate,12,2),3.);
    end;
    else do;
        shr = 0;
        ehr = 0;
    end;

    if minlevel ge 5 then do;                       *minutes;
        smin = input(substr(&sdate,15,2),3.);
        emin = input(substr(&edate,15,2),3.);
    end;
    else do;
        smin = 0;
        emin = 0;
    end;

    if minlevel = 6 then do;                        *seconds;
        ssec = input(substr(&sdate,18,2),3.);
        esec = input(substr(&edate,18,2),3.);
    end;
    else do;
        ssec = 0;
        esec = 0;
    end;


%mend minlevel;
```

%DURATION MACRO:

```
%macro duration(sdate,edate,outcome);

   /* call %minlevel */
   %minlevel;

    if minlevel in(1,2) then &outcome = trim(left(put(years,2.))) || 'y';
    if minlevel = 2 then &outcome = &outcome || trim(left(put(months,2.))) || 'm';

    /* calculate each piece of duration */
    if minlevel ge 3 then do;
        sdttm = dhms(sday,shr,smin,ssec);
        edttm = dhms(eday,ehr,emin,esec);

        secdur = edttm - sdttm;

        if secdur ne 0 then do;
          secs = mod(secdur,60);

          mindur = (secdur - secs)/60;
          mins = mod(mindur,60);

          hrdur = (mindur - mins)/60;
          hours = mod(hrdur,24);

          daydur = (hrdur - hours)/24;
          days = mod(daydur,7);

          wkdur = (daydur - days)/7;
          weeks = mod(wkdur,52);

          years = (wkdur - weeks)/52;

          if (years gt 0) then &outcome = trim(left(put(years,3.))) || 'y';
           if (weeks gt 0) then &outcome = left(trim(left(&outcome)) || trim(left(put(weeks,z2.))) ||
                                           'w');
           if (days gt 0)  then &outcome = left(trim(left(&outcome)) || trim(left(put(days,z2.))) ||
                                           'd');
          if (hours gt 0) then &outcome = left(trim(left(&outcome)) || trim(left(put(hours,z2.))) ||
                                           'h');
          if (mins gt 0)  then &outcome = left(trim(left(&outcome)) || trim(left(put(mins,z2.)))  ||
                                           'm');
          if (secs gt 0)  then &outcome = left(trim(left(&outcome)) || trim(left(put(secs,z2.)))  ||
                                           's');
        end;
        else do;
               select(minlevel);
                   when(3) &outcome = '01d';
                   when(4) &outcome = '01h';
                   when(5) &outcome = '01m;
                   when(6) &outcome = '01s';
               end;
        end;
    end;

    if minlevel = . then do;
           &outcome = '';
           if (minlevel ne 0) then put 'WARN''ING:(ACI) Not enough information to calculate duration. '
                                       usubjid= &sdate= &edate= minlevel=;
    end;


%mend duration;
```

%STUDYDAY MACRO:

```
%macro studyday(sdate,edate,outcome);

   %minlevel;

    /* calculate the study day based on the derivation rule */
    if (minlevel ge 3) then do;
        if eday ge sday then &outcome = eday - sday + 1;
        else &outcome = eday - sday;
    end;
    else do;
        &outcome = .;
        if (minlevel ne 0) then put 'WARN' 'ING:(ACI) Not enough information to calculate study day: '
                                   usubjid= &sdate= &edate= minlevel=;
    end;

%mend studyday;
```

%TEAE MACRO:

```
%macro teae(outcome,datevar,srefdate,erefdate=,lag=0);

       /* compare AE start date with reference start date first */
       %let sdate=&srefdate;
       %let edate=&datevar;
       %minlevel;

       /* compare each available piece to determine TEAE */
       if (minlevel le 2) then do;
              if (eyr lt syr) then &outcome = 'N';
              else if (minlevel eq 2 and eyr eq syr) then do;
                     if (emon lt smon) then &outcome = 'N';
             end;
       end;
       else if (minlevel ge 3) then do;
              if (eday lt sday) then &outcome = 'N';
              else if (minlevel ge 4) and (eday eq sday) then do;
                     if (ehr lt shr) then &outcome = 'N';
                     else if (minlevel ge 5) and (ehr eq shr) then do;
                            if (emin lt smin) then &outcome = 'N';
                            else if (minlevel eq 6) and (emin eq smin) then do;
                                   if (esec lt ssec) then &outcome = 'N';
                            end;
                     end;
              end;
       end;

       /* if reference end date exists, compare AE start date with reference end date */
      %if &erefdate ne  %then %do;

              %let sdate=&datevar;
              %let edate=&erefdate;

              %minlevel;

              /* when both dates are at least full date, then the lag time can be easily
                 added and compare the dates;
               */
             if (minlevel ge 3) then do;
                     /* add the lag time */
                     eday = eday + &lag;

                     if (eday lt sday) then &outcome = 'N';
                     else if (minlevel ge 4) and (eday eq sday) then do;
                            if (ehr lt shr) then &outcome = 'N';
                            else if (minlevel ge 5) and (ehr eq shr) then do;
                                   if (emin lt smin) then &outcome = 'N';
                                   else if (minlevel eq 6) and (emin eq smin) then do;
                                          if (esec lt ssec) then &outcome = 'N';
                                   end;
                            end;
                     end;
              end;
       else do;
                /* when sdate is partial, but edate still full date, we can still compare;
                   since date already get each piece in the minlevel, we may use the variable created
                    in the macro to get the end date in days, */
                if (level2 ge 3) then do;
                  enday  = input(substr(&edate, 1, 10), yymmdd10.);
                  /* add the lag time*/
                  enday  = enday + &lag;
                  /* put the end date back to level 2, since here only compare with partial AE start
                     date*/
                  cendt  = substr(put(enday, yymmdd10.), 1, 7);
                  /* keep the new month part of end date for comparison */
                  endmon = input(scan(cendt,2,'-'),best.);

                  /* compare with partial AE start date */
                  if (level1 = 1) then do;
                     if (eyr le syr) then &outcome = 'N';
                  end;
```

```
                  else if (level1 = 2) and (eyr eq syr) then do;
                              if (endmon lt smon) then &outcome = 'N';
                  end;
                end;
                else do;
                       /*if end date also partial, unless the lag=0, otherwise,
                         we can't determine it*/
                       if &lag=0 then do;
                          if (minlevel le 2) then do;
                             if (eyr lt syr) then &outcome = 'N';
                             else if (minlevel eq 2) and (eyr eq syr) then do;
                                     if (emon lt smon) then &outcome = 'N';
                              end;
                            end;
                        end;
                end;
          end;
    %end;

       /* No WARN ING needed since the emergence is only set if it
          can be proven the event occurred outside set period */
        if &outcome ne 'N' then &outcome = '';

%mend teae;
```