Module to create Global Macro Variables with distinctive names matching Date values they carry.

Anatoly V. Kulinsky, Holland, PA

ABSTRACT

Macro Function is designed to generate various uniquely named Global variables to hold date values corresponding to the names. Macro variables created automatically for future use in SAS code. Available arguments include date formats and time intervals, alignment within the intervals, explicit starting point or default system date, number and direction of iterations – up to seven parameters with assigned default values. This flexibility allows using same tool with different arguments to produce multiple macro variables of different date format representing essentially same date, if requires.

The INTNX function is used to shift time interval. CALL SYMPUT routine uses date values to create macro variables. Implementation allows referencing and comparing dates stored in the variables or embedded into various SAS names, external file or database names. It could be employed to create time-series suffixes or custom formats allowing automation and summarization of data by applying combination of two different date formats on the same data.

Function is designed as an application for users. Error Handling introduced to exit program using %GOTO and %LABEL statements with explanatory ERROR messages to the SAS LOG achieving self-tutorial effect. It is equipped for job control and produces Return Codes permitting to identify even multiple exceptions. This function is written for SAS v.8 both UNIX and Windows environment.

KEYWORDS:

Macro function date format INTNX time-series error handling return code

INTRODUCTION

Many routine tasks performed in Financial and Pharmaceutical industries depend on date values. Data usually collected over certain period, on a range of visits, within either one interval or spanning over a series. For data analyses time dependent comparisons are often essential. Data could be structured as multi-record transactions, or organized horizontally as one record with multiple variables that differs by ‘time-stamp’ suffix, or summarized on account level in cycles or as of month-end. Most of the original data comes as daily files. Those carry information on customers, their behavior, and transactions such as balance transfers, cash advances, purchases, and returns. Most financial performance data for instance cash and merchandise APR, outstanding balances, financial charges and fees, cost of funds, income and so on are updated on monthly basis. To further complicate the issue some of the most important information from Credit Bureaus comes on the quarterly basis.

%MACRO ‘ANYDATE’ was developed to automate data pull from Oracle DB and to create SAS Data Mart from individual datasets. Global macro variables were used to reference date dimension either embedded into table name or using ‘date’ variables on indexed/partitioned tables with ‘WHERE’ statement. Further automation and flexibility is achieved by introduction of several arguments allowing producing various return values – macro date variables.

GOAL

Generate various Global Macro variables representing date in requested format for any required number of iterations for future use in SAS code.

RATIONALE:

1. Function has to create varied Global Macro variables with unique names holding matching date values automatically so it could be used anywhere in SAS code without overriding. At the same time it has to allow for customization. To be used for/diverse date formats and intervals, alignment within the interval. Adjustable start point, number of iterations – subsequently requires Macro with keyword parameters assigned default values.
2. Representation of date formats and accordingly corresponding Macro variable names should be conventional for both SAS programmers as well as occasional users and at the same time helpful in straightforward visualization like `YMMY`, `MMDDYY` or `YYMMDD`.

3. It should not go beyond the scope of the specified utility tool so that trying to solve any possible problem like ‘spell-out’ date format would not diminish its usefulness. Further development of the corresponding but separate tools with calls to the present one will be done.

4. It should be user friendly. Frequently, motives to read and implement new code are either to learn from it or in hope to make life easier. To help achieve later, elaborate Error Handling was introduced to exit program using `%GOTO` and `%LABEL` statements with `ERROR / WARNING` explanatory messages whenever some sort of program exception is come across. An additional benefit is self-tutorial effect. Just use it!

ASSUMPTIONS

1. The user aware of how the `INTNX` function works and notion of interval boundaries.

2. Date value of Macro variables been created and it’s name defines by the argument Format. Key-letters convention – ‘Y’ defines year; ‘Q’ defines quarter; ‘M’ defines month; ‘W’ defines week and ‘D’ defines day. Any `Format` that includes string ‘Date’ defines regular SAS `DATEw. Format`. ‘Julian’ as well as ‘Jul’ or ‘DDD’ or ‘Day’ - `JULIANw. Format`. ‘MMM’ or ‘MON’ defines `MonYYw. or YYMonw. Formats`. No special characters allowed.

3. `Quantity` of key-letters like ‘Y’ or ‘M’ defines total requested `Format` width – length of value of Macro variables. Arguments `Interval` and `Format` (and `Startdt`) should not be mismatched!

4. Name of the first – ‘Start Date’ global macro variable been created have attached suffix ‘0’, in all additional variable names suffix represents order of iteration, with next most recent having ‘1’, next ‘2’ and so on.

5. Underscore between created variables’ name and suffix signals that requested points in time-series are going backward, absence of it – onward.

TECHNICAL DETAILS

SAS dates are special numbers stored as integers. Dates before January 1, 1960 are negative integers and after January 1, 1960 are positive. However, we are used to different representation of time, no one can easily tell what date is 15796 or –21549. To use them we need to see customary dates. We habitually use formats typically: `DATE`, `MMDDYY` or `YYMMDD`. Forms of these formats are consistent and self-explanatory but format width ‘`w`’– not always. It is not easy to remember how ‘03MAR03’d will be written out by `MMDDYY5.` – as 33103 or 03/31 or 03-03. Most convenient formats for data processing purposes probably are `YYMMn6.` or `YYMMDDn8.` In them ‘n’ stands for no separator, thus avoiding inconsistency and introduction of any special characters. Arguably, date representation should be in order from highest date unit to the lowest, i.e. from years to days. Placing year first followed by month allows easy sort and comparison of any dates in both numeric and character forms, eliminates national differences.

Function `TODAY()` is the source of the default start date unless you need specific or constant date. It was chosen over more intuitive automatic macro variable `$SYSDATE` due to present ambiguous behavior of it in SAS version 8.x under Windows environment. Argument “`Startdt`” could be specified instead. CALL SYMPUT routine uses date values to create requested number of global macro variables each representing date in specified format.

In the foundation is SAS date function `INTNX`. Function `INTNX` generates SAS date value that is shifted by specified type and number of intervals from the start. Produced result depends on interval boundaries. Each time interval aligned to the beginning within this interval, by default. For example: interval ‘Month’ was requested and starting date specified as ‘26MAY2004’. Interval ‘Month’ aligned date to the first day. The actual start date will be ’01MAY2004’. Interval ‘Week’ for the same date will start from ’23MAY2004’ – Sunday. Alignment could be assigned to the middle or to the end of the requested period also.

`INTNX` syntax is: `INTNX (`interval`, start-from, increment `<`, ‘alignment’`).

For example:

```sas
mydate = intnx('month', '26MAY2004'd , -1, 'm');
```

Will assign value 16176 to the variable mydate – i.e. ‘05/19/2004’ in `mmddyy10. Format`.

Further, generating macro variable and assigning date value to it:

```sas
CALL SYMPUT('yymmdd', put(mydate, yymmdd6.));
```

Macro variable named “yymmdd” holds value ‘040415’.
%MACRO ‘ANYDATE’ is designed to accommodate a variety of possible factors, build SAS code dynamically based on supplied arguments. It has seven keyword parameters. Six have default values. They are: (1) "Format" to represent desired date format and accordingly created Macro variables name and to evaluate width for ’w’ value, (2) "Interval" – time Interval (step) like month or week, (3) "Increment" to request actual starting point be apart from the start–from point for number of steps, (4) "Align" – alignment to the beginning, middle or the end of the period and letters 'b', 'm', 'e' could be used as abbreviation., (5) "First" – iterations start point of additional date variables, (6) "Last" – iterations end point (number of iterations). The last parameter – (7) "Startdt" has no default value and should be specified if hard-coded fixed start date is desirable.

Macro Function reads input parameters to achieve a few objectives: produce both Macro variables and their date values in requested format. Control for correct input parameter values and type, special characters, mismatch between format and start date, if used. Module assigns proper format width and converts user format into regular SAS date format. Adds correct index - suffix to the variable name based on required direction and number of iterations. Underscore between created variables’ name and suffix signals that requested points are going backward in time, absence of it – onward.

Macro Module handles apparent disparity between format and interval, issuing either WARNING message – explaining the potential problem since deviation could be intentional or ERROR message. ERROR terminates program execution using %GOTO and %LABEL statements with detailed, descriptive error message in the Log achieving self-tutorial effect. At the same time each ERROR produces unique Error Return Code identifying as single as multiple exceptions. Utilization of SAS LOG and Error Return Codes permits control job execution due to complexity and potential variations of all function’s arguments for errors or for their mutual incompatibility, thus helping end-users with various skill levels.

In addition Function produces descriptive Notes - auto messages to the Log as references to both program flow and the results. See Examples.

**USAGE OF MACRO FUNCTION**

1. Automation of data pull either by table name or using ‘date’ variables on indexed/partitioned tables with ’where’ statement.
2. Automation of data processing by referencing and comparing date values embedded into SAS libname, dataset or variable name; external directory/file/field or database schema/table/column name.
   a). In Set/Merge used in data set or variable names.
   b). Direct or conditional variable processing using both names and/or the values including use in the SAS functions.
   c). Embedding formatted dates into various SAS and non-SAS names. SAS names modification creating time-series type suffixes.
3. Creating custom formats allowing automation and summarization of data by applying combination of two different date formats on the same data.
4. Reporting and representation purposes, creating constants.

**CONCLUSION**

1. Automatically generates various Global Macro variables representing date in requested format for any required number of iterations with unique names holding matching date values automatically for use anywhere in SAS code without overriding it.
2. Accommodates a variety of possible factors, builds SAS code dynamically based on supplied arguments.
3. User friendly. SAS LOG is utilized to control job execution with explanatory messages. Error Handling introduced to test all function’s arguments for errors or for their mutual incompatibility and either reassign the value or terminate program execution using %GOTO and %LABEL statements therefore allows self-tutorial effect. ERROR produces unique Error Return code identifying as single as multiple exceptions thus allow automation of the following usage programs as well.
4. Automates data pull either by referencing table name or using ‘date’ variables on indexed/partitioned tables with ‘where’ statement.
5. Automation of data processing by referencing and comparing date values embedded into SAS libname, dataset or variable name; external directory/file/field or database schema/table/column name.
6. Creating custom formats allowing automation and summarization of data by applying combination of two different date formats on the same data.
7. Automation of data reporting and representation, creating constants.
REFERENCES


TRADEMARKS

The SAS System® is a registered trademark of SAS Institute, Inc., Cary, NC.

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.

CONTACT INFORMATION

Anatoly V. Kulinsky
571 Sugartree Rd.
Holland PA 18966
Phonc: (215) 357-7001

SAS LOG Example 1:

*******************************************************************************
*** FORMAT "yymmdd" and INTERVAL "w" are valid!***
*******************************************************************************
* WARNING: INTERVAL "w" and FORMAT "yymmdd" are DISSIMILAR!*  
* WARNING: DATE will be shifted by Interval’s ALIGNMENT to be represented correctly.*
*******************************************************************************
* Direction and sign for INTNX function is "+" and Increment is "0".*
*******************************************************************************
>Start date for INTNX function is "20040221" with length 8 bytes. Format used is yymmddn6.*
*******************************************************************************
>Start date used to create requested Macro Variables is sysd=16116 sysd=15FEB2004 sysd=02/15/2004
*******************************************************************************
> Generating Macro Var named yymmdd0 for format=040215*
> Generating Macro Var named yymmdd1 for format=040222*
> Generating Macro Var named yymmdd2 for format=040229*
> Generating Macro Var named yymmdd3 for format=040307*
*******************************************************************************
* Dates for requested Interval "week" in format "yymmdd" are:
* SAS format "yymmddn6" was used. Date "040215" is first created.
* 3 additional "weeks" generated: "040307" through "040222".
*******************************************************************************
### Example SAS LOG Example 2:

```
*** Error Return Codes: retrn_cd=0* sum_rc=0*
> displ>Value0 requested: <0> MacroVar0=ymmd0=040215*
> displ>Value1 requested: <1> MacroVar1=ymmd1=040222*
> displ>Value2 requested: <2> MacroVar2=ymmd2=040229*
> displ>Value3 requested: <3> MacroVar3=ymmd3=040307*
```

```
Example SAS LOG Example 2:

 ************************************************
*** FORMAT "julian7" and INTERVAL "d" are valid!*
************************************************
* Direction and sign for INTNX function is "-" and Increment is "0".*
*********************************************************************
>Start date for INTNX function is "2004011" with length 7 bytes. Format used is julian7.*
*****************************************************************************************
>Start date used to create requested Macro Variables is sysd=16081 sysd=11JAN2004 sysd=01/11/2004
************************************************************************************************************
> Generating Macro Var named juliane0 for format0 =2004011*
> Generating Macro Var named juliane_1 for format_1=2004010*
> Generating Macro Var named juliane_2 for format_2=2004009*
> Generating Macro Var named juliane_3 for format_3=2004008*
****************************************************************
* Dates for requested Interval "day" in format "juliane" are:
* SAS format "julian7" was used. Date "2004011" is last created. 
* 3 additional "days" generated: "2004008" through "2004010".
****************************************************************
*** Error Return Codes: retrn_cd=0* sum_rc=0*
> displ>Value0 requested: <0> MacroVar0=juliane0=2004011*
> displ>Value1 requested: <1> MacroVar1=juliane_1=2004010*
> displ>Value2 requested: <2> MacroVar2=juliane_2=2004009*
> displ>Value3 requested: <3> MacroVar3=juliane_3=2004008*
```

### EXAMPLE OF MACRO USE:

**EXAMPLE 1**

/* Creating SAS libname, data sets and variables with embedded in name date stamp from Oracle tables with date in the name. */

```
%include "&dmdir/modules/anydate.mod";
&anydate(format=yyyymo, interval=m, incremnt=-1, last=6);
&anydate(format=ymo, interval=m, incremnt=-1, last=6);
&anydate(format=monyr, interval=m, incremnt=-1, last=6);

LIBNAME curr "&maindir/&monyr";

%IF %EVAL(&last >= &first) %THEN
%DO;
   %DO m = &first %TO &last;
   %put ** > Currently creating *&pgmname.&&yrmo&m* Data Set from *XXXX_&&yyyrmo&m* Table **;
   %let n=&m;
   PROC SQL;
   CONNECT TO ORACLE (USER=&oralogin ORAPW=&orapaswd PATH="@XXX" buffsize=4096);
   CREATE table curr.&pgmname.&&yrmo&m as
   SELECT
      PID length=6 informat=11. label="PID &&yrmo" 
      , PAYMT&&yrmo&m length=6  informat=9.2  label="Payment Amount &&yrmo&m"
      , ENDBAL&&yrmo&m length=6  informat=9.2 label=" End Bal &&yrmo&m"
      , PROFIT_SEG   PDTAG&&yrmo&m length=3  informat=3. label="Profit Driver Tag &&yrmo&m"
   FROM CONNECTION TO ORACLE
   (SELECT
      PID 
      , nvl(PAY_AMT,0) PAYMT&&yrmo&m 
      , nvl(END_BAL,0) ENDBAL&&yrmo&m 
      , PROFIT_SEG PDTAG&&yrmo&m 
      FROM XXX.XXX_&&yyyrmo&m)
   ;
   DISCONNECT from ORACLE;
%END;
QUIT;
```
EXAMPLE 2

* Call of macro with two date formats;

```sas
%include "&dmdir/modules/anydate.mod";
%anydate(format=yyyrmo, interval=m, increment=-1, last=6);
%anydate(format=yrmo, interval=m, increment=-1, last=6);
```

Control message to log:

```sas
%put>displ>Macro Variables/Formats requested:
<0> MacroVarName = &displ = &&&displ *;
<1> MacroVarName1 = &displ1 = &&&displ1 *;
<2> MacroVarName2 = &displ2 = &&&displ2 *;
<3> MacroVarName3 = &displ3 = &&&displ3 *;
<4> MacroVarName4 = &displ4 = &&&displ4 *;
<5>MacroVarName5 = &displ5 = &&&displ5 *;
<6>MacroVarName6 = &displ6 = &&&displ6 *;
```

```sas
%MACRO Pull_Data(mo_back=0, first=1, last=11);
  %LOCAL m mo_back first last;
  %IF %EVAL(&last >= &first) %THEN
    %DO;
      %DO m = &first %TO &last;
        %put>Creating m=&m* ds=&dsname.&&yrmo&m* Data Set from XXX Table EXT_DT=&&yyyrmo&m*;
        %put> MO_PYMT_AMT&m=MO_PYMT_AMT%EVAL(&last+1-&m)*yrmo&m=&&yrmo&m* corresponding to %EVAL(&last+1-&m)*;
        PROC SQL;
          CONNECT TO ORACLE (USER=&oralogin ORAPW=&orapaswd PATH="@XXX" buffsize=1000);
          CREATE table &dsname.&&yrmo&m AS
          SELECT
          PID                 length=6 informat=11. format=11. LABEL='PERSON ID'
          , CUR_BAL%EVAL(&last+1-&m) length=6 informat=9.
          , LS_LT_CHRG%EVAL(&last+1-&m) length=4 informat=6.
          , LS_CSH_INT%EVAL(&last+1-&m) length=4 informat=6.
          , MO_CSH_AMT%EVAL(&last+1-&m) length=5 informat=8.
          , MO_PUR_AMT%EVAL(&last+1-&m) length=5 informat=8.
          , MO_PMT_AMT%EVAL(&last+1-&m) length=5 informat=8.
          , CY_CSH_AMT length=5  informat=8. LABEL="CASH AMOUNT"
          , LS_DATE length=4 informat=6.
          FROM CONNECTION TO ORACLE
          (SELECT
          PID
          , CUR_BAL CUR_BAL%EVAL(&last+1-&m)
          , LS_LT_CHRG LS_LT_CHRG%EVAL(&last+1-&m)
          , LS_CSH_INT LS_CSH_INT%EVAL(&last+1-&m)
          , MO_CSH_AMT MO_CSH_AMT%EVAL(&last+1-&m)
          , MO_PUR_AMT MO_PUR_AMT%EVAL(&last+1-&m)
          , CY_CSH_AMT CY_CSH_AMT%EVAL(&last+1-&m)
          , LS_DATE
          FROM XXX.XXXXXXXXXXX
          WHERE &criter and EXT_DT=&&yyyrmo&m);
        QUIT;
        PROC SORT data=&dsname.&&yrmo&m nodupkey ;
        by pid ;
        RUN;
    %END;
    %put>&dsname.&&yrmo&m last created. Generating &m DS: &dsname.&&yrmo&m* through &dsname.&&yrmo&m*;
  %END;
  %ELSE %DO ;
    %put>Data Set &dsname.&&yrmo&m is the last created. There are no extra Data Sets generated: Last less than First! ***;
  %END;
%MEND Pull_Data;
%Pull_Data(mo_back = 0, first = 1, last = 6);
%put yrmo1=&&yrmo1* yrmo2=&&yrmo2* yrmo3=&&yrmo3* yrmo4=&&yrmo4* yrmo5=&&yrmo5* yrmo6=&&yrmo6*;
```
DATA curr.&dsname;      * Combine all data sets *;
  MERGE
    &dsname.&yrmo1(in=last)
    &dsname.&yrmo2
    &dsname.&yrmo3
    &dsname.&yrmo4
    &dsname.&yrmo5
    &dsname.&yrmo6;
  BY pid;
  IF last;
    IF LS_DATE < &yrmo6 THEN DELETE;
RUN;

EXAMPLE 3
  * Call of essentially same macro within other macro for 9 consecutive month;

%MACRO Spin(start = 2);
  %let i = %EVAL(&start+8);
  %DO n = &start %TO &i;
    %let j = &n;
    %Pull_Data(first = &j, last = %EVAL(&j+1));
  %END;
  %put> Data Set ds=&dsname*;
%MEND Spin;

%Spin(start = 2);
DATA ma.xxx;  * Combine all data sets *;
  SET
    maq.v200110
    maq.v200111
    maq.v200112
    maq.v200201
    maq.v200202
    maq.v200203
    maq.v200204
    maq.v200205
    maq.v200206;
RUN;

PROC SUMMARY DATA = ma.xxx nway;
  CLASS YR_QTR MOB;
  VAR xxx_CNT xxyy;
  OUTPUT OUT=ma.xxx_summ(drop=_type_) sum= ;
RUN;

EXAMPLE 4
  /* To summarize variable by applying date formats. Does not require to change
dates manually. Last date to be included must be at least 1-year back */
%include "&dmdir/modules/anydate.mod";
%anydate(format=yyyrmodd, interval=mm, incremnt=-1, align=b, last=22);
%anydate(format=monyr, interval=mm, incremnt=-1, align=b, last=22);

PROC FORMAT ;
  VALUE $ wavefmt
    low="&yyyrmodd_22"="Pre-&monyr_22"
    "&yyyrmodd_19"="&monyr_22.-&monyr_19"
    "&yyyrmodd_16"="&monyr_19.-&monyr_16"
    "&yyyrmodd_13"="&monyr_16.-&monyr_13"
    "&yyyrmodd_11"="&monyr_13.-&monyr_11"
    "&yyyrmodd_11"="&monyr_11.-&monyr_11";
RUN;

DATA a;
  SET nt.learn_lab(where=(MAX_FLAP_EXP>0));
    MAX_FLAP_EXP_c = put(MAX_FLAP_EXP,8.); * converting to char dt to use with fmt ;
RUN;
PROC FREQ data=a;
  TABLES MAX_FLAP_EXP_c / list ;
  FORMAT MAX_FLAP_EXP_c $wavefmt.;
RUN;