

More Fun With a Saturday Nite Date

Deb Cassidy, Cardinal Health Inc.

ABSTRACT

For some reason, almost every presentation I've seen about SAS® dates used some catchy title. Maybe it's because working with calendar days isn't the most thrilling SAS code to write but it is often very essential code.

This paper will cover a variety of examples for working with SAS dates and times - both basic and advanced. It will include some situations I recently encountered such as the following. How do you know it's a 'Saturday nite date' versus a weekday so having a '0' value for sales is OK? How do you write a production job to keep a rolling 18-months of data? How do you write a production job that will check the date of the incoming monthly file and only run when it is the expected month? How do you merge two files based on date and time when the times are not identical? How do you create a SAS date from a partial date?

All code shown here is available in Version 6.10 under Windows. Some may not be available in earlier versions or other operating systems. HELP for date-related statements is more detailed in Version 6.11.

BASICS

Why does SAS need a special date value? You could store dates as character variables but you can only report the dates. You could store dates as numbers such as 19961030 which the computer would equate to 19,961,030. In this case, you could also compare dates. For example, the date could show when a customer invoice was generated. If the dates were 19961028 and 19961030 then you could subtract and know the second invoice was 2 days after the first. But what if the dates were 19961028 and 19961101? Subtracting gives you 73 days when the invoice was only 3 days later. You could write a complicated program to calculate the correct value but SAS has already taken care of that for you. Using SAS dates also permit you to print dates in calendar order regardless of whether you start with year, month, day or day-of-week.

SAS dates are the number of days since January 1, 1960. I've been assured that there was a very logical reason for that date - it wasn't to make some of us seem old because our birthday is negative. Any date prior to January 1, 1960 will be a negative value. Here are a few examples:

<u>Date</u>	<u>SAS Date</u>
January 1, 1582	-138061
January 1, 1900	-21914
September 30, 1956	-1188
December 1, 1959	-1
January 1, 1960	0
January 2, 1960	1
September 30, 1996	13422
February 29, 2000	14669
February 29, 2001	missing
October 16, 2233	100000
December 31, 20000	6589335

As you can tell, SAS accounts for leap years. If you request a non-existent date, you'll get a missing value. The first and last dates shown indicate the range of possible dates.

SAS also have a time value which is the number of seconds after midnight. You can also have a date/time value which is the number of seconds from midnight on January 1, 1960. Time can be reported in fractions of a second.

To specify a constant value for a date, you simply use 'DDMONYY'd or 'DDMONYYYY'd where DD is the 2-digit day, MON is the 3-character abbreviation for month and YY or YYYY is the year. If you only use the 2-digit year, SAS will assume the century based on the value set in the yearcutoff option. By default, this is 1900 but your SAS administrator may have changed it.

The yearcutoff option controls the beginning of the 100-year period which is assumed if you do not specify a century. When the cutoff is 1900, then all years are assumed to be 1900-1999 when you don't have the century specified. Note that date values are based upon the yearcutoff that was in effect when the value was created and not when it is printed. This will enable you to merge or set datasets that were created with different

Yearcutoff values. However, if you will be using multiple values, be sure to use a format that is wide enough to print the century. The following table shows the results of using different cutoff values.

Yearcutoff	01JAN00	01JAN60	31DEC99
1900	01JAN1900	01JAN1960	31DEC1999
1920	01JAN2000	01JAN1960	31DEC1999
1960	01JAN2000	01JAN1960	31DEC1999
1961	01JAN2000	01JAN2060	31DEC1999
1999	01JAN2000	01JAN2060	31DEC1999
2000	01JAN2000	01JAN2060	31DEC2099

To specify a constant time, use 'HH:MM't or 'HH:MM:SS't or 'HH:MM:SS.SS't. Values are based on a 24-hour clock. To specify a constant date/ time value, combine the two with a colon such as 'DDMONYY:HH:MM'dt. A time of 12:65 is legitimate but you'll discover it prints as 13:05 by moving any minutes over 59 to the next hour. Times are not checked for validity so you can have 999:999:999 as a legitimate time.

READING IN DATE VALUES

There are several informat to let you read in date or time values. These are:

<u>INFORMAT</u>	<u>WIDTH</u>	<u>DESCRIPTION</u>
DATEw	7-32	ddmonyy
DDMMYYw	6-32	ddmmyy
JULIANw	5-32	yyddd
MMDDYYw	6-32	mmddyy
YYMMDDw	6-32	yyymmdd
MONYYw	5-32	monyy
YYQw	4-32	yyq
TIMEw	5-32	hh:mm:ss.s
DATETIMEw.d	13-40	ddmmyy:hh:mm:ss.ss

In the above table, YY, MM, DD and Q are the numeric values for year, month, day, and quarter, respectively. MON is the 3-character abbreviation for month. HH,MM,SS.SS are hours, minutes and seconds. The last format combines date and time. There are also some special formats for records from an IBM mainframe.

Why would you ever need a width of 32? SAS is very lenient about spaces and other characters in dates. The following is valid code while the value 1196 is not. However, such code isn't recommended. For example, 11896 is

interpreted as Jan 18, 1996 but what if you really wanted November 8, 1996?

```
data one;
  infile cards;
  input date mmddyy32.;
cards;
(start-up) 1 18,1999
(end) 10 30 2001
month 11 day 29 year 1956
01181996
011796
11896
011796 12
;
```

But what do you do if your date is in the form of MONDDYY or MONTHDDYY where MONTH is the full name of the month? You'll need to read in each part of the date as a separate field and then use the MDY function to create the date. If the values are not valid input for the MDY function, you will also need to convert them to valid values first.

For example, my data is in the form: September 30, 1996. I create individual variables so I have month='September', day='30' and year='1996'. The statement would be:

```
DATE=MDY(MONTH, DAY, YEAR);
```

Unfortunately, you'll discover that 'September' is not valid input nor is 'Sep'. However, if you change the value to '09', the function works. You will get a message that character values have been converted to numeric. You can change the names of the months to numbers by using IF statements or creating a custom informat. You should also try to read the day and year as numeric so the character-numeric conversion does not occur.

What about a partial date? For example, my data from the mainframe is in the form YYYYMM. I can still create a date by assuming the value of day. For convenience, the day is often assumed to be 1. Depending on the analysis you are doing, you may wish to assume it's the 15th or write code to determine the 'middle' or 'end' of the month which will vary based on your month. What if month or year is missing, too? You can also make assumptions about those values but you must consider how you will be using the data.

Here's another case: Your dataset already exists with dates of YYYYMM and YYYYMMDD as numeric values so you can't read year, month and day individually. In my case, my data is coming in as packed decimal so I must read it as one numeric value. However, using some other functions, you can create a date from the existing numeric value. The easier case is when you have only month and year so you will assume the day is the first of the month.

```
ORIGDATE=199609;
DATE=MDY (
    MOD (ORIGDATE, 100) ,
    1,
    INT (ORIGDATE/100) );
```

The INT function returns the integer portion of the value. The MOD function returns the remainder. The year is the integer portion of 199609 / 100 and the month is the remainder of that division. Note that the divisor is one of the parameters for the MOD function but the INT function uses the division statement. The INT function can also be used on values such as 1996.09

The code is a little more complex when you have a YYYYMMDD value.

```
ORIGDATE=19960930;
DATE=MDY (
    MOD (INT (ORIGDATE/100) , 100) ,
    MOD (ORIGDATE, 100) ,
    INT (ORIGDATE/10000) );
```

To get the year portion of the date, simply divide 19960930 by 10000. The result is 1996.093 and the integer portion is 1996. The day portion is the remainder when you divide 19960930 by 100. The month is in the middle so it's a little trickier. First, eliminate the day portion by taking the integer portion of 19960930/100. That value is 199609. Now the month will be the remainder when you divide the new value by 100.

But what if your date is a text value? That's actually easier because you just need to substring the value.

```
ORIGDATE='19960930';
DATE=MDY (
    SUBSTR (ORIGDATE, 5, 2) ,
    SUBSTR (ORIGDATE, 7, 2) ,
    SUBSTR (ORIGDATE, 1, 4) );
```

PRINTING WITH FORMATS

The table at the end of the paper shows a variety of formats. All the formats can be printed using different widths. The table shows the results of the minimum, maximum and default widths for all the formats listed. The 'RESULT' header shows 40 characters with the '+' as 32 which is the most common maximum width for the formats.

As you look over the examples, you should notice that parts of the date or time are missing if you use a small width. Make sure your widths are wide enough especially if you have dates from different centuries. SAS will print as much of the value as possible. If your width is wider than necessary, the result will be right-justified. Notice the datetime format didn't give the correct result because the original value wasn't a date/time value.

Are you wondering about those formats at the end of the date section? SAS also lets you print some of the formats in several different languages. Currently available languages and their abbreviations are:

<u>Language</u>	<u>Abbreviation</u>
French	FRA
Italian	ITA
German	DEU
Danish	DAN
Finnish	FIN
Spanish	ESP
Swedish	SVE
Norwegian	NOR
Dutch	NLD
Portuguese	PTG
Swiss_German	DES
Swiss_French	FRS

If you will only work in one language, you can specify that language with

```
OPTIONS DFLANG=language;
```

The default value is English unless your system administrator changed it. You can then use formats and informats similar to EURDFWKX. If you want to use different languages, you'll need to change the EUR to the appropriate abbreviation. Please note that not all date/time

formats and informats are available in all the languages.

What if you want a format that isn't available? For example, you want MM-DD-YYYY but the MMDDYY format only prints with slashes and not dashes. You can solve this by creating a new variable that is customized. There are functions which return the individual month, day and year values. Simply concatenate the individual values with your separation symbol as shown below.

```
NEWDATE=MONTH(ORIGDATE) || '-' ||  
          DAY(ORIGDATE) || '-' ||  
          YEAR (ORIGDATE);
```

Your results are:

9- 30- 1996

which obviously isn't what you want. The functions return numeric values which are then converted to character values. Lead blanks are included since numeric values are right-justified. To get around the problem, simply force the numeric values to the appropriate length. One way is:

```
NEWDATE=PUT(MONTH(ORIGDATE), Z2.) || '-' ||  
          PUT(DAY(ORIGDATE), Z2.) || '-' ||  
          PUT(YEAR(ORIGDATE), Z4.);
```

The result is 09-30-1996. The Zx format puts lead 0's on numeric values to force the value the specified width. Since year is already a 4-digit number, using Z2 there would give you '***'.

MODIFYING THE FORMAT OF RUN DATE

By default, your output will have a date/time stamp in the upper-right-hand corner of the page in the form:

17:03 Monday, June 24, 1996

But what if you don't want the time or you want a title that says:

DATA AS OF 01SEP96

This is very easy to do. Use OPTIONS NODATE to suppress the default date/time and use the following as your title.

TITLE "DATA AS OF &SYSDATE";

You must use double quotes to have the macro variable be resolved. &SYSDATE is one of several automatic macro variables available to you. You don't need to be within a macro to use it.

What if you want the date in a different format? That is still fairly easy but it can't be done directly. The following code will create a new macro variable which is the formatted value. You can not use &SYSDATE in the code but the TODAY function and the DATE function will both give you the current date based on the computer's clock. You simply create a new variable by formatting the result of the function. This value is then moved into a macro variable using CALL SYMPUT. Note the macro variable name must be in quotes because it is the name of the macro variable you are creating.

```
DATA _NULL_ ;  
TODAY=PUT(TODAY(), YMMDD.);  
CALL SYMPUT('RUNDATE', TODAY);  
RUN;  
TITLE "DATA AS OF &RUNDATE";
```

IS IT A SATURDAY?

The title for this paper came about because I needed to know if certain days were a Saturday or Sunday because our company doesn't always receive sales from our divisions on those days. There are several ways to show this. One is to print the dates using the WEEKDATE, WEEKDATX or DOWNAME formats. If you specify any width from 3-8, you will get the 3-character abbreviation for the day. If you specify a width of 9, you will get the full name of the day. However, DOWNAME will result in right-justified names. In all cases, the first letter is capital and the rest are lower-case. For WEEKDATE and WEEKDATX, you can specify a width of 14 and still return just the name of the day.

What if you want the names in uppercase or you want to label Saturday and Sunday as 'WEEKEND'? What if you want to be able to put weekends in a different dataset? The WEEKDAY function can be used to create a new variable which will have the day of the week. You can then create a custom format for this variable. WEEKDAY returns a 1 for Sunday and 7 for Saturday.

```
DATA WEEKENDS WEEK; SET MYDATA;
  DAY=WEEKDAY(ORIGDATE);
  IF DAY IN (1,7) THEN OUTPUT WEEKENDS;
  ELSE OUTPUT WEEK;
```

```
PROC FORMAT ;
  VALUE DAYNAME
    1,7='WEEKEND'
    2='MON'   3='TUES'
    4='WED'   5='THURS'
    6='FRI';
PROC PRINT DATA=WEEK;
  FORMAT DAY DAYNAME.;
```

CREATING A DATE WHEN NONE IS GIVEN

On the first day of the month, I receive a file that is a snapshot of the file as of the end of the previous month. I summarize the file and append it to the existing summarized file. However, there is no date field so I must create one. I want the date to represent the previous month and not the date the job ran so I can't use the TODAY() or DATE() functions. Actually, I can but in conjunction with another function.

INTNX will advance a date by the specified time period. For example,

```
ORIGDATE='28SEP96'D;
NEWDATE1=INTNX('DAY',ORIGDATE,1);
NEWDATE2=INTNX('DAY',ORIGDATE,20);
NEWDATE3=INTNX('DAY',ORIGDATE,-5);
```

will set NEWDATE1 to 29SEP96, NEWDATE2 to 18OCT96 and NEWDATE3 to 23SEP96. The latter case 'advances' a negative amount so it 'reverses'.

You can advance by DAY, WEEK, MONTH, QUARTER or YEAR. However, only DAY advances by a specific number of days. WEEK advances to the beginning of the week (Sunday) rather than 7 days. MONTH, QUARTER and YEAR will advance to the first day of the month, quarter and year, respectively. Here are some examples for the original date of 28SEP96.

PERIOD	#	NEW DATE
WEEK	1	29SEP96
WEEK	10	01DEC96
MONTH	1	01OCT96
MONTH	12	01SEP97
MONTH	-1	01AUG96
MONTH	0	01SEP96
QUARTER	1	01OCT96

QUARTER	3	01APR97
YEAR	1	01JAN97
YEAR	-1	01JAN95

Whenever you have a Saturday date, advancing by DAY or by WEEK will give you the same result.

For my situation, advancing by a -1 month was the easiest solution. End-users will only specify a month and year so the program can be easily written to use the first of the relevant month instead of the last day which can vary from 28-31. My code is then:

```
SKU_DTE=INTNX('MONTH',TODAY(),-1);
```

KEEPING A ROLLING RANGE OF MONTHS

I have a monthly summary file and I need to always keep 18 months of data. Date represents the first day of the month. The following code is similar to the above and will delete my 'old' month when my 'new' month is added. This job always runs after a month ends so my September run will be for August data. This is one of those times when you should thoroughly test your code before you really delete the records to verify you are keeping the correct months.

```
IF ACSUDATE<INTNX('MONTH',TODAY(),-18)
  THEN DELETE;
```

CHECKING INCOMING DATES

How do you write a production job that will check the date of the incoming monthly file and only run when it is the expected month? In my case, a file was created on the mainframe which required the computer operators to specify the month. A job on our Unix system would FTP the file and append the data to a SAS dataset. Since most of the operators were new employees, mistakes happened. We also had no idea when the file would be created and old files were not deleted from the mainframe. The Unix system was set up to run this job at a particular time.

To avoid problems, we took advantage of a Unix feature that will only run an 'automatic' job if a specified file is in place. First, another 2-line file was created when the main file was created and

FTP'd directly to Unix. The first line was a message indicating the main file was created and the second line contained the date of the data, the date and time it ran and the number of records. A SAS job would check this file daily and create another file when the date was correct. The FTP job would only run when this 'OK to run' file was found. The code to check the dates is shown below. The macros are necessary to be able to conditionally execute the DATA _NULL_ step to create the file. I could also have used the X statement to issue a Unix command that will create an empty file.

```
DATA CHECK;
  INFILE '/FTP/ACSUSEND.TXT';
  INPUT @1 FILEDTE $6.;
  IF FILEDTE='AN ACS' THEN DELETE;
  INPUTDTE=MOY(SUBSTR(FILEDTE,5,2) ,
    '01' , SUBSTR(FILEDTE,1,4) );
  REQMONTH=INTNX('MONTH',TODAY(),-1);
  CALL SYMPUT('INPUTDTE',INPUTDTE);
  CALL SYMPUT('REQMONTH',REQMONTH);

%MACRO WRITEFILE;
  DATA _NULL_;
  SET CHECK;
  FILE '/SASDATA/RUNACSU.TXT';
  PUT INPUTDTE DATE7.;
  RUN;
%MEND WRITEFILE;

%MACRO CHKDATE;
  %PUT 'THIS IS INPUT====>' &INPUTDTE;
  %PUT 'THIS IS REQUEST==>' &REQMONTH;
  %IF &INPUTDTE=&REQMONTH %THEN %WRITEFILE;
%MEND CHKDATE;
%CHKDATE;
```

HOW MUCH TIME HAS PASSED?

If you need to know how many weeks or months have passed then you need the INTCK function. For example, this paper was due July 1, 1996 (duedate) and the conference started September 29, 1996 (confstrt).

```
WEEKS=('WEEK',DUEDATE,CONFSTART);
MONTHS=('MONTH',DUEDATE,CONFSTR);
```

The answer is 13 weeks or 2 months. This function counts the number of interval beginnings and doesn't include partial intervals. Since July 1 was a Monday, you'll find 13 Sundays are encountered to reach September 29. August and September were the only months that started. You can have date intervals of day, week, month, qtr and year along with several datetime and time intervals.

MATCHING FUZZY DATES

How do you merge two files based on date or time when you don't have exact matches? The following code will match records from dataset two which are 2 days before or after the records in dataset one.

```
DATA ONE;
  X='03JAN96'D;OUTPUT;
  X='06JAN96'D; OUTPUT;

DATA TWO;
  Y='01JAN96'D;OUTPUT;
  Y='11JAN96'D; OUTPUT;

PROC SQL;
CREATE TABLE CHECK AS
  SELECT *
  FROM ONE, TWO
  WHERE Y BETWEEN X-2 AND X+2;

PROC PRINT DATA=CHECK;
  FORMAT X Y DATE7.;
```

The results are:

OBS	X	Y
1	03JAN96	01JAN96

OTHER USEFUL FUNCTIONS

Do you have a date/time value but you want the date and time to be separate variables?

```
DATE=DATEPART(ORIGINAL);
TIME=TIMEPART(ORIGINAL);
```

Do you need to convert to a Julian date?

```
NEWDATE=DATEJUL(ORIGINAL);
```

You can also retrieve a numeric value for quarter, hour, minute or second just like you can for month, day and year. For the time functions, your original value can be a time or a datetime value.

```
QTR_NUM=QTR(ORIGDATE);
HOURS=HOUR(ORIGTIME);
MINUTES=MINUTE(ORIGTIME);
SECONDS=SECOND(ORIGTIME);
```

If you know the year and quarter, you can create a SAS date which will be the first day of the quarter. In this second example, YEAR is a 2 or 4-digit year and quarter is either 1,2,3 or 4.

```
QTR_YR=YYQ(YEAR,QTRNUM);
```

The current system time can also be returned. If you are running an interactive session, it will be the current time and not the time you started the session. The latter time appears in the default date/time heading on your output. There is also an automatic macro variable &SYSTIME.

```
CURRENT=TIME();
```

You can create a SAS time value from individual values of variables. You must have a value for seconds although it can be 0. Examples are:

```
NEWTIME=HMS(HOUR,MINUTE,SECOND);  
NEWTIME2=HMS(5,12,00);  
NEWTIME3=HMS(5,65,00);
```

Do you need to know how many days are in a month? You can combine several functions into one statement. The resulting value can then be used in a

```
LASTDAY=DAY((INTNX('MONTH',X,1) -1));
```

How do you know if the system administrator changed a default such as DFLANG or YEARCUTOFF? PROC OPTIONS will print all the current settings for options. Any options you've already over-ridden in your session will have the new value. Even though this is a PROC, the results will appear in your log rather than your output.

SUMMARY

This paper provides a variety of things to do with dates. Many of these examples came about from writing production code that requires little or no human intervention regardless of whether the data are available at the scheduled time.

Due to space limitations, a few examples are not included here. These examples will be distributed at the presentation or may be obtained later from the author.

TRADEMARK

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

REFERENCES

SAS Institute Inc., *SAS® Language: Reference, Version 6, First Edition* Cary, NC: SAS Institute Inc., 1990. 1042 pp.

SAS Institute Inc., *SAS® Guide to the SQL Procedure: Usage and Reference, Version 6, First Edition* Cary, NC: SAS Institute Inc., 1989. 210 pp.

CONTACT INFO

Deb Cassidy
Cardinal Health Inc
5555 Glendon Court
Dublin, OH 43016
(614) 717-7136
dcassidy@cardhealth.com

FORMAT EXAMPLES FOR THE DATE SEPTEMBER 30, 1996

FORMAT	TYPE	-----RESULT-----+
DATE5.	MIN	30SEP
DATE6.		30SEP
DATE7.	DEFAULT	30SEP96
DATE8.		30SEP96
DATE9.	MAX	30SEP1996
DDMMYY2.	MIN	30
DDMMYY4.		3009
DDMMYY5.		30/09
DDMMYY8.	DEFAULT	30/09/96
DDMMYY10.	MAX	30/09/1996
MMDDYY2.	MIN	09
MMDDYY8.	DEFAULT	09/30/96
MMDDYY10.	MAX	09/30/1996
YYMMDD2.	MIN	96
YYMMDD8.	DEFAULT	96-09-30
YYMMDD10.	MAX	1996-09-30
WEEKDATE3.	MIN	Mon
WEEKDATE29.	DEFAULT	Monday, September 30, 1996
WEEKDATE37.	MAX	Monday, September 30, 1996
WEEKDATX3.	MIN	Mon
WEEKDATX29.	DEFAULT	Monday, 30 September 1996
WEEKDATX37.	MAX	Monday, 30 September 1996
WORDDATE3.	MIN	Sep
WORDDATE18.	DEFAULT	September 30, 1996
WORDDATE32.	MAX	September 30, 1996
WORDDATX3.	MIN	Sep
WORDDATX19.	DEFAULT	30 September 1996
WORDDATX32.	MAX	30 September 1996
JULIAN5.	MIN/DEFAULT	96274
JULIAN7.	MAX	1996274
MONYY5.	MIN/DEFAULT	SEP96
MONYY7.	MAX	SEP1996
YYMON5.	MIN	96SEP
YYMON7.	DEFAULT	1996SEP
YYMON32.	MAX	1996SEP
YYMM5.	MIN	96M09
YYMM7.	DEFAULT	1996M09
YYMM32.	MAX	1996M09

YYMMd5.	MIN	96-09	
YYMMd7.	DEFAULT	1996-09	
YYMMd32.	MAX		1996-09
YYMMc5.	MIN	96:09	
YYMMc7.	DEFAULT	1996:09	
YYMMc32.	MAX		1996:09
YYMMn5.	MIN	9609	
YYMMn7.	DEFAULT	199609	
YYMMn32.	MAX		199609
YYMMp5.	MIN	96.09	
YYMMp7.	DEFAULT	1996.09	
YYMMp32.	MAX		1996.09
YYMms5.	MIN	96/09	
YYMms7.	DEFAULT	1996/09	
YYMms32.	MAX		1996/09
MMYY5.	MIN	09M96	
MMYY7.	DEFAULT	09M1996	
MMYY32.	MAX		09M1996
YEAR2.	MIN	96	
YEAR4.	DEFAULT	1996	
YEAR32.	MAX		1996
MONTH2.	MIN/DEFAULT	9	
MONTH32.	MAX		9
DAY2.	MIN/DEFAULT	30	
DAY32.	MAX		30
DOWNNAME1.	MIN	M	
DOWNNAME9.	DEFAULT	Monday	
DOWNNAME32.	MAX		Monday
WEEKDAY1.	MIN/DEFAULT	2	
WEEKDAY32.	MAX		2
MONNAME1.	MIN	S	
MONNAME9.	DEFAULT	September	
MONNAME32.	MAX		September
JULDAY3.	MIN/DEFAULT	274	
JULDAY32.	MAX		274
YYQ4.	MIN	96Q3	
YYQ6.	DEFAULT/MAX	1996Q3	
YYQR6.	MIN	96QIII	
YYQR8.	DEFAULT	1996QIII	
YYQR32.	MAX		1996QIII
YYQC4.	MIN	96:3	
YYQC6.	DEFAULT	1996:3	

YYQC32.	MAX		1996:3
YYQRC6.	MIN	96:III	
YYQRC8.	DEFAULT	1996:III	
YYQRC32.	MAX		1996:III
(YYQx and YYQRx use the separators as YYMMx)			
QTR1.	MIN/DEFAULT	3	
QTR32.	MAX		3
QTRR3.	MIN/DEFAULT	III	
QTRR32.	MAX		III
DATETIME7.	MIN	01JAN60	
DATETIME16.	DEFAULT	01JAN60:03:43:42	
DATETIME40.	MAX		01JAN1960:03:43:42
NENG02.	MIN	08	
NENG010.	MAX/DEFAULT	H.08/09/30	
FRADFWKX.	DEFAULT	Lundi 30 septembre 1996	
DANDFWKX.	DEFAULT	mandag, den 30. september 1996	
NLDDFWKX.	DEFAULT	maandag, 30 september 1996	
FINDFWKX.	DEFAULT	Maanantaina, 30. syyskuuta 1996	
DEUDFWKX.	DEFAULT	Montag, 30. September 1996	
ITADFWKX.	DEFAULT	Lunedì, 30 Settembre 1996	
NORDFWKX.	DEFAULT	mandag, 30 september 1996	
PTGDFWKX.	DEFAULT	Segunda-feira, 30 de Setembro de 1996	
ESPDFWKX.	DEFAULT	lunes, 30 de septiembre de 1996	
SVEDFWKX.	DEFAULT	måndag, 30 september 1996	
FRSDFWKX.	DEFAULT	Lundi 30 septembre 1996	
DESDFWKX.	DEFAULT	Montag, 30. September 1996	

FORMAT EXAMPLES FOR THE TIME 11:30 AM

TIME2.	MIN	11	
TIME8.	DEFAULT	11:30:00	
TIME20.	MAX		11:30:00
HHMM2.	MIN	11	
HHMM5.	DEFAULT	11:30	
HHMM20.	MAX		11:30
MMSS2.	MIN	**	
MMSS5.	DEFAULT	690	
MMSS20.	MAX		690:00
HOUR2.	MIN/DEFAULT	12	
HOUR20.	MAX		12
TOD2.	MIN	11	
TOD8.	DEFAULT	11:30:00	
TOD20.	MAX		11:30:00
DATETIME7.	MIN	01JAN60	
DATETIME16.	DEFAULT	01JAN60:11:30:00	
DATETIME40.	MAX		01JAN1960:11:30:00