Title

stata.com

Date and time functions

Contents Functions Video example Reference Also see

Contents

dow(e_d)	the numeric day of the week corresponding to date e_d; 0 = Sunday, 1 = Monday, ..., 6 = Saturday
doy(e_d)	the numeric day of the year corresponding to date e_d
halfyear(e_d)	the numeric half of the year corresponding to date e_d
halfyearly(s_1,s_2[,Y])	the e_h half-yearly date (half-years since 1960h1) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see date()
hh(e_{tc})	the hour corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
hhC(e_{tC})	the hour corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
hms(h,m,s)	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to h, m, s on 01jan1960
hofd(e_d)	the e_h half-yearly date (half years since 1960h1) containing date e_d
hours(ms)	ms/3,600,000
mdy(M,D,Y)	the e_d date (days since 01jan1960) corresponding to M, D, Y
mdyhms(M,D,Y,h,m,s)	the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to M, D, Y, h, m, s
minutes(ms)	ms/60,000
mm(e_{tc})	the minute corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
mmC(e_{tC})	the minute corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
mofd(e_d)	the e_m monthly date (months since 1960m1) containing date e_d
month(e_d)	the numeric month corresponding to date e_d
monthly(s_1,s_2[,Y])	the e_m monthly date (months since 1960m1) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see date()
msofhours(h)	$h \times$ 3,600,000
msofminutes(m)	$m \times$ 60,000
msofseconds(s)	$s \times$ 1,000
qofd(e_d)	the e_q quarterly date (quarters since 1960q1) containing date e_d
quarter(e_d)	the numeric quarter of the year corresponding to date e_d
quarterly(s_1,s_2[,Y])	the e_q quarterly date (quarters since 1960q1) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see date()
seconds(ms)	ms/1,000
ss(e_{tc})	the second corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
ssC(e_{tC})	the second corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
tC(l)	convenience function to make typing dates and times in expressions easier
tc(l)	convenience function to make typing dates and times in expressions easier
td(l)	convenience function to make typing dates in expressions easier
th(l)	convenience function to make typing half-yearly dates in expressions easier

`tm(`l`)`	convenience function to make typing monthly dates in expressions easier
`tq(`l`)`	convenience function to make typing quarterly dates in expressions easier
`tw(`l`)`	convenience function to make typing weekly dates in expressions easier
`week(`e_d`)`	the numeric week of the year corresponding to date e_d, the `%td` encoded date (days since 01jan1960)
`weekly(`s_1`,`s_2`[,`Y`])`	the e_w weekly date (weeks since 1960w1) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see `date()`
`wofd(`e_d`)`	the e_w weekly date (weeks since 1960w1) containing date e_d
`year(`e_d`)`	the numeric year corresponding to date e_d
`yearly(`s_1`,`s_2`[,`Y`])`	the e_y yearly date (year) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see `date()`
`yh(`Y`,`H`)`	the e_h half-yearly date (half-years since 1960h1) corresponding to year Y, half-year H
`ym(`Y`,`M`)`	the e_m monthly date (months since 1960m1) corresponding to year Y, month M
`yofd(`e_d`)`	the e_y yearly date (year) containing date e_d
`yq(`Y`,`Q`)`	the e_q quarterly date (quarters since 1960q1) corresponding to year Y, quarter Q
`yw(`Y`,`W`)`	the e_w weekly date (weeks since 1960w1) corresponding to year Y, week W

Functions

Stata's date and time functions are described with examples in **[U] 24 Working with dates and times** and [D] **datetime**. What follows is a technical description. We use the following notation:

e_b	`%tb` business calendar date (days)
e_{tc}	`%tc` encoded datetime (ms. since 01jan1960 00:00:00.000)
e_{tC}	`%tC` encoded datetime (ms. with leap seconds since 01jan1960 00:00:00.000)
e_d	`%td` encoded date (days since 01jan1960)
e_w	`%tw` encoded weekly date (weeks since 1960w1)
e_m	`%tm` encoded monthly date (months since 1960m1)
e_q	`%tq` encoded quarterly date (quarters since 1960q1)
e_h	`%th` encoded half-yearly date (half-years since 1960h1)
e_y	`%ty` encoded yearly date (years)
M	month, 1–12
D	day of month, 1–31
Y	year, 0100–9999
h	hour, 0–23
m	minute, 0–59
s	second, 0–59 or 60 if leap seconds
W	week number, 1–52
Q	quarter number, 1–4
H	half-year number, 1 or 2

The date and time functions, where integer arguments are required, allow noninteger values and use the `floor()` of the value.

A Stata date-and-time (`%t`) variable is recorded as the milliseconds, days, weeks, etc., depending upon the units from 01jan1960; negative values indicate dates and times before 01jan1960. Allowable dates and times are those between 01jan0100 and 31dec9999, inclusive, but all functions are based on the Gregorian calendar, and values do not correspond to historical dates before Friday, 15oct1582.

`bofd("`*cal*`",`e_d`)`
- Description: the e_b business date corresponding to e_d
- Domain *cal*: business calendar names and formats
- Domain e_d: `%td` as defined by business calendar named *cal*
- Range: as defined by business calendar named *cal*

`Cdhms(`e_d`,`h`,`m`,`s`)`
- Description: the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to e_d, h, m, s
- Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
- Domain h: integers 0 to 23
- Domain m: integers 0 to 59
- Domain s: reals 0.000 to 60.999
- Range: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers −58,695,840,000,000 to >253,717,919,999,999) or *missing*

`Chms(`h`,`m`,`s`)`
- Description: the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to h, m, s on 01jan1960
- Domain h: integers 0 to 23
- Domain m: integers 0 to 59
- Domain s: reals 0.000 to 60.999
- Range: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers −58,695,840,000,000 to >253,717,919,999,999) or *missing*

`Clock(`s_1`,`s_2`[,`Y`])`
- Description: the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to s_1 based on s_2 and Y

  Function `Clock()` works the same as function `clock()` except that `Clock()` returns a leap second–adjusted `%tC` value rather than an unadjusted `%tc` value. Use `Clock()` only if original time values have been adjusted for leap seconds.
- Domain s_1: strings
- Domain s_2: strings
- Domain Y: integers 1000 to 9998 (but probably 2001 to 2099)
- Range: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers −58,695,840,000,000 to >253,717,919,999,999) or *missing*

`clock(`s_1`,`s_2`[ ,`Y`])`

Description: the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to s_1 based on s_2 and Y

s_1 contains the date, time, or both, recorded as a string, in virtually any format. Months can be spelled out, abbreviated (to three characters), or indicated as numbers; years can include or exclude the century; blanks and punctuation are allowed.

s_2 is any permutation of `M`, `D`, [*##*]`Y`, `h`, `m`, and `s`, with their order defining the order that month, day, year, hour, minute, and second occur (and whether they occur) in s_1. *##*, if specified, indicates the default century for two-digit years in s_1. For instance, s_2 = `"MD19Y hm"` would translate s_1 = `"11/15/91 21:14"` as 15nov1991 21:14. The space in `"MD19Y hm"` was not significant and the string would have translated just as well with `"MD19Yhm"`.

Y provides an alternate way of handling two-digit years. Y specifies the largest year that is to be returned when a two-digit year is encountered; see function `date()` below. If neither *##* nor Y is specified, `clock()` returns *missing* when it encounters a two-digit year.

Domain s_1: strings
Domain s_2: strings
Domain Y: integers 1000 to 9998 (but probably 2001 to 2099)
Range: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers −58,695,840,000,000 to 253,717,919,999,999) or *missing*

`Cmdyhms(`M`,`D`,`Y`,`h`,`m`,`s`)`

Description: the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) corresponding to M, D, Y, h, m, s
Domain M: integers 1 to 12
Domain D: integers 1 to 31
Domain Y: integers 0100 to 9999 (but probably 1800 to 2100)
Domain h: integers 0 to 23
Domain m: integers 0 to 59
Domain s: reals 0.000 to 60.999
Range: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers −58,695,840,000,000 to >253,717,919,999,999) or *missing*

`Cofc(`e_{tc}`)`

Description: the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of e_{tc} (ms. without leap seconds since 01jan1960 00:00:00.000)
Domain e_{tc}: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers −58,695,840,000,000 to 253,717,919,999,999)
Range: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers −58,695,840,000,000 to >253,717,919,999,999)

`cofC(`e_{tC}`)`

Description: the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of e_{tc} (ms. without leap seconds since 01jan1960 00:00:00.000)
Domain e_{tC}: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers −58,695,840,000,000 to >253,717,919,999,999)
Range datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers −58,695,840,000,000 to 253,717,919,999,999)

`Cofd(`e_d`)`
Description: the e_{tC} datetime (ms. with leap seconds since 01jan1960 00:00:00.000) of date e_d at time 00:00:00.000
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers −58,695,840,000,000 to >253,717,919,999,999)

`cofd(`e_d`)`
Description: the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) of date e_d at time 00:00:00.000
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers −58,695,840,000,000 to 253,717,919,999,999)

`daily(`s_1`,`s_2`[,`Y`])`
Description: a synonym for `date(`s_1`,`s_2`[,`Y`])`

`date(`s_1`,`s_2`[,`Y`])`
Description: the e_d date (days since 01jan1960) corresponding to s_1 based on s_2 and Y

s_1 contains the date, recorded as a string, in virtually any format. Months can be spelled out, abbreviated (to three characters), or indicated as numbers; years can include or exclude the century; blanks and punctuation are allowed.

s_2 is any permutation of `M`, `D`, and [*##*]Y, with their order defining the order that month, day, and year occur in s_1. *##*, if specified, indicates the default century for two-digit years in s_1. For instance, s_2 = `"MD19Y"` would translate s_1 = `"11/15/91"` as 15nov1991.

Y provides an alternate way of handling two-digit years. When a two-digit year is encountered, the largest year, *topyear*, that does not exceed Y is returned.

```
date("1/15/08","MDY",1999) = 15jan1908
date("1/15/08","MDY",2019) = 15jan2008

date("1/15/51","MDY",2000) = 15jan1951
date("1/15/50","MDY",2000) = 15jan1950
date("1/15/49","MDY",2000) = 15jan1949

date("1/15/01","MDY",2050) = 15jan2001
date("1/15/00","MDY",2050) = 15jan2000
```

If neither *##* nor Y is specified, `date()` returns *missing* when it encounters a two-digit year. See *Working with two-digit years* in [D] **datetime translation** for more information.
Domain s_1: strings
Domain s_2: strings
Domain Y: integers 1000 to 9998 (but probably 2001 to 2099)
Range: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549) or *missing*

`day(`e_d`)`
Description: the numeric day of the month corresponding to e_d
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: integers 1 to 31 or *missing*

`dhms(`e_d`,`h`,`m`,`s`)`
Description: the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to e_d, h, m, and s
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Domain h: integers 0 to 23
Domain m: integers 0 to 59
Domain s: reals 0.000 to 59.999
Range: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers −58,695,840,000,000 to 253,717,919,999,999) or *missing*

`dofb(`e_b`,"`cal`")`
Description: the e_d datetime corresponding to e_b
Domain e_b: `%tb` as defined by business calendar named cal
Domain cal: business calendar names and formats
Range: as defined by business calendar named cal

`dofC(`e_{tC}`)`
Description: the e_d date (days since 01jan1960) of datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
Domain e_{tC}: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers −58,695,840,000,000 to >253,717,919,999,999)
Range: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)

`dofc(`e_{tc}`)`
Description: the e_d date (days since 01jan1960) of datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
Domain e_{tc}: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers −58,695,840,000,000 to 253,717,919,999,999)
Range: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)

`dofh(`e_h`)`
Description: the e_d date (days since 01jan1960) of the start of half-year e_h
Domain e_h: `%th` dates 0100h1 to 9999h2 (integers −3,720 to 16,079)
Range: `%td` dates 01jan0100 to 01jul9999 (integers −679,350 to 2,936,366)

`dofm(`e_m`)`
Description: the e_d date (days since 01jan1960) of the start of month e_m
Domain e_m: `%tm` dates 0100m1 to 9999m12 (integers −22,320 to 96,479)
Range: `%td` dates 01jan0100 to 01dec9999 (integers −679,350 to 2,936,519)

`dofq(`e_q`)`
Description: the e_d date (days since 01jan1960) of the start of quarter e_q
Domain e_q: `%tq` dates 0100q1 to 9999q4 (integers −7,440 to 32,159)
Range: `%td` dates 01jan0100 to 01oct9999 (integers −679,350 to 2,936,458)

`dofw(`e_w`)`
Description: the e_d date (days since 01jan1960) of the start of week e_w
Domain e_w: `%tw` dates 0100w1 to 9999w52 (integers −96,720 to 418,079)
Range: `%td` dates 01jan0100 to 24dec9999 (integers −679,350 to 2,936,542)

`dofy(`e_y`)`
Description: the e_d date (days since 01jan1960) of 01jan in year e_y
Domain e_y: `%ty` dates 0100 to 9999 (integers 0100 to 9999)
Range: `%td` dates 01jan0100 to 01jan9999 (integers −679,350 to 2,936,185)

`dow(`e_d`)`
Description: the numeric day of the week corresponding to date e_d; 0 = Sunday, 1 = Monday, ..., 6 = Saturday
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: integers 0 to 6 or *missing*

`doy(`e_d`)`
Description: the numeric day of the year corresponding to date e_d
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: integers 1 to 366 or *missing*

`halfyear(`e_d`)`
Description: the numeric half of the year corresponding to date e_d
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: integers 1, 2, or *missing*

`halfyearly(`s_1`,`s_2`[,`Y`])`
Description: the e_h half-yearly date (half-years since 1960h1) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see `date()`
Domain s_1: strings
Domain s_2: strings `"HY"` and `"YH"`; `Y` may be prefixed with `##`
Domain Y: integers 1000 to 9998 (but probably 2001 to 2099)
Range: `%th` dates 0100h1 to 9999h2 (integers −3,720 to 16,079) or *missing*

`hh(`e_{tc}`)`
Description: the hour corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
Domain e_{tc}: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers −58,695,840,000,000 to 253,717,919,999,999)
Range: integers 0 through 23, *missing*

`hhC(`e_{tC}`)`
Description: the hour corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
Domain e_{tC}: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999 (integers −58,695,840,000,000 to >253,717,919,999,999)
Range: integers 0 through 23, *missing*

`hms(`h`,`m`,`s`)`
Description: the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to h, m, s on 01jan1960
Domain h: integers 0 to 23
Domain m: integers 0 to 59
Domain s: reals 0.000 to 59.999
Range: datetimes 01jan1960 00:00:00.000 to 01jan1960 23:59:59.999 (integers 0 to 86,399,999 or *missing*)

`hofd(`e_d`)`
Description: the e_h half-yearly date (half years since 1960h1) containing date e_d
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: `%th` dates 0100h1 to 9999h2 (integers −3,720 to 16,079)

`hours(`ms`)`
Description: ms/3,600,000
Domain ms: real; milliseconds
Range: real or *missing*

`mdy(`M`,`D`,`Y`)`
Description: the e_d date (days since 01jan1960) corresponding to M, D, Y
Domain M: integers 1 to 12
Domain D: integers 1 to 31
Domain Y: integers 0100 to 9999 (but probably 1800 to 2100)
Range: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549) or *missing*

`mdyhms(`M`,`D`,`Y`,`h`,`m`,`s`)`
Description: the e_{tc} datetime (ms. since 01jan1960 00:00:00.000) corresponding to M, D, Y, h, m, s
Domain M: integers 1 to 12
Domain D: integers 1 to 31
Domain Y: integers 0100 to 9999 (but probably 1800 to 2100)
Domain h: integers 0 to 23
Domain m: integers 0 to 59
Domain s: reals 0.000 to 59.999
Range: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers −58,695,840,000,000 to 253,717,919,999,999) or *missing*

`minutes(`ms`)`
Description: ms/60,000
Domain ms: real; milliseconds
Range: real or *missing*

`mm(`e_{tc}`)`
Description: the minute corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
Domain e_{tc}: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers −58,695,840,000,000 to 253,717,919,999,999)
Range: integers 0 through 59, *missing*

`mmC(`e_{tC}`)`
Description: the minute corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
Domain e_{tC}: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers −58,695,840,000,000 to >253,717,919,999,999)
Range: integers 0 through 59, *missing*

`mofd(`e_d`)`
Description: the e_m monthly date (months since 1960m1) containing date e_d
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: `%tm` dates 0100m1 to 9999m12 (integers −22,320 to 96,479)

`month(`e_d`)`
Description: the numeric month corresponding to date e_d
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: integers 1 to 12 or *missing*

`monthly(`s_1`,`s_2`[ ,`Y`])`
Description: the e_m monthly date (months since 1960m1) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see `date()`
Domain s_1: strings
Domain s_2: strings `"MY"` and `"YM"`; `Y` may be prefixed with ##
Domain Y: integers 1000 to 9998 (but probably 2001 to 2099)
Range: `%tm` dates 0100m1 to 9999m12 (integers −22,320 to 96,479) or *missing*

`msofhours(`h`)`
Description: $h \times 3{,}600{,}000$
Domain h: real; hours
Range: real or *missing*; milliseconds

`msofminutes(`m`)`
Description: $m \times 60{,}000$
Domain m: real; minutes
Range: real or *missing*; milliseconds

`msofseconds(`s`)`
Description: $s \times 1{,}000$
Domain s: real; seconds
Range: real or *missing*; milliseconds

`qofd(`e_d`)`
Description: the e_q quarterly date (quarters since 1960q1) containing date e_d
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: `%tq` dates 0100q1 to 9999q4 (integers −7,440 to 32,159)

`quarter(`e_d`)`
Description: the numeric quarter of the year corresponding to date e_d
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: integers 1 to 4 or *missing*

`quarterly(`s_1`,`s_2`[ ,`Y`])`
Description: the e_q quarterly date (quarters since 1960q1) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see `date()`
Domain s_1: strings
Domain s_2: strings `"QY"` and `"YQ"`; `Y` may be prefixed with ##
Domain Y: integers 1000 to 9998 (but probably 2001 to 2099)
Range: `%tq` dates 0100q1 to 9999q4 (integers −7,440 to 32,159) or *missing*

`seconds(`ms`)`
Description: $ms/1{,}000$
Domain ms: real; milliseconds
Range: real or *missing*

`ss(`e_{tc}`)`
Description: the second corresponding to datetime e_{tc} (ms. since 01jan1960 00:00:00.000)
Domain e_{tc}: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers −58,695,840,000,000 to 253,717,919,999,999)
Range: real 0.000 through 59.999, *missing*

`ssC(`e_{tC}`)`
Description: the second corresponding to datetime e_{tC} (ms. with leap seconds since 01jan1960 00:00:00.000)
Domain e_{tC}: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers −58,695,840,000,000 to >253,717,919,999,999)
Range: real 0.000 through 60.999, *missing*

`tC(`l`)`
Description: convenience function to make typing dates and times in expressions easier

Same as `tc()`, except returns leap second–adjusted values; for example, typing `tc(29nov2007 9:15)` is equivalent to typing `1511946900000`, whereas `tC(29nov2007 9:15)` is `1511946923000`.
Domain l: datetime literal strings 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
Range: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers −58,695,840,000,000 to >253,717,919,999,999)

`tc(`l`)`
Description: convenience function to make typing dates and times in expressions easier

For example, typing `tc(2jan1960 13:42)` is equivalent to typing `135720000`; the date but not the time may be omitted, and then 01jan1960 is assumed; the seconds portion of the time may be omitted and is assumed to be 0.000; `tc(11:02)` is equivalent to typing `39720000`.
Domain l: datetime literal strings 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
Range: datetimes 01jan0100 00:00:00.000 to 31dec9999 23:59:59.999
(integers −58,695,840,000,000 to 253,717,919,999,999)

`td(`l`)`
Description: convenience function to make typing dates in expressions easier

For example, typing `td(2jan1960)` is equivalent to typing `1`.
Domain l: date literal strings 01jan0100 to 31dec9999
Range: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)

`th(`l`)`
Description: convenience function to make typing half-yearly dates in expressions easier

For example, typing `th(1960h2)` is equivalent to typing `1`.
Domain l: half-year literal strings 0100h1 to 9999h2
Range: `%th` dates 0100h1 to 9999h2 (integers −3,720 to 16,079)

`tm(`l`)`
Description: convenience function to make typing monthly dates in expressions easier

For example, typing `tm(1960m2)` is equivalent to typing `1`.
Domain l: month literal strings 0100m1 to 9999m12
Range: `%tm` dates 0100m1 to 9999m12 (integers −22,320 to 96,479)

`tq(`l`)`
Description: convenience function to make typing quarterly dates in expressions easier

For example, typing `tq(1960q2)` is equivalent to typing `1`.
Domain l: quarter literal strings 0100q1 to 9999q4
Range: `%tq` dates 0100q1 to 9999q4 (integers −7,440 to 32,159)

`tw(`l`)`
Description: convenience function to make typing weekly dates in expressions easier

For example, typing `tw(1960w2)` is equivalent to typing `1`.
Domain l: week literal strings 0100w1 to 9999w52
Range: `%tw` dates 0100w1 to 9999w52 (integers −96,720 to 418,079)

`week(`e_d`)`
Description: the numeric week of the year corresponding to date e_d, the `%td` encoded date (days since 01jan1960)

Note: The first week of a year is the first 7-day period of the year.
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range integers 1 to 52 or *missing*

`weekly(`s_1`,`s_2`[,`Y`])`
Description: the e_w weekly date (weeks since 1960w1) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see `date()`
Domain s_1: strings
Domain s_2: strings `"WY"` and `"YW"`; `Y` may be prefixed with *##*
Domain Y: integers 1000 to 9998 (but probably 2001 to 2099)
Range: `%tw` dates 0100w1 to 9999w52 (integers −96,720 to 418,079) or *missing*

`wofd(`e_d`)`
Description: the e_w weekly date (weeks since 1960w1) containing date e_d
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: `%tw` dates 0100w1 to 9999w52 (integers −96,720 to 418,079)

`year(`e_d`)`
Description: the numeric year corresponding to date e_d
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: integers 0100 to 9999 (but probably 1800 to 2100)

`yearly(`s_1`,`s_2`[,`Y`])`
Description: the e_y yearly date (year) corresponding to s_1 based on s_2 and Y; Y specifies *topyear*; see `date()`
Domain s_1: strings
Domain s_2: string `"Y"`; `Y` may be prefixed with *##*
Domain Y: integers 1000 to 9998 (but probably 2001 to 2099)
Range: `%ty` dates 0100 to 9999 (integers 0100 to 9999) or *missing*

`yh(`Y`,`H`)`
Description: the e_h half-yearly date (half-years since 1960h1) corresponding to year Y, half-year H
Domain Y: integers 1000 to 9999 (but probably 1800 to 2100)
Domain H: integers 1, 2
Range: `%th` dates 1000h1 to 9999h2 (integers −1,920 to 16,079)

`ym(`Y`,`M`)`
Description: the e_m monthly date (months since 1960m1) corresponding to year Y, month M
Domain Y: integers 1000 to 9999 (but probably 1800 to 2100)
Domain M: integers 1 to 12
Range: `%tm` dates 1000m1 to 9999m12 (integers −11,520 to 96,479)

`yofd(`e_d`)`
Description: the e_y yearly date (year) containing date e_d
Domain e_d: `%td` dates 01jan0100 to 31dec9999 (integers −679,350 to 2,936,549)
Range: `%ty` dates 0100 to 9999 (integers 0100 to 9999)

`yq(`Y`,`Q`)`
Description: the e_q quarterly date (quarters since 1960q1) corresponding to year Y, quarter Q
Domain Y: integers 1000 to 9999 (but probably 1800 to 2100)
Domain Q: integers 1 to 4
Range: `%tq` dates 1000q1 to 9999q4 (integers −3,840 to 32,159)

`yw(`Y`,`W`)`
Description: the e_w weekly date (weeks since 1960w1) corresponding to year Y, week W
Domain Y: integers 1000 to 9999 (but probably 1800 to 2100)
Domain W: integers 1 to 52
Range: `%tw` dates 1000w1 to 9999w52 (integers −49,920 to 418,079)

Video example

How to create a date variable from a date stored as a string

Reference

Rajbhandari, A. 2015. A tour of datetime in Stata. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2015/12/17/a-tour-of-datetime-in-stata-i/.

Also see

[FN] **Functions by category**

[D] **datetime** — Date and time values and variables

[D] **egen** — Extensions to generate

[D] **generate** — Create or change contents of variable

[M-5] **date( )** — Date and time manipulation

[U] **13.3 Functions**