
```

THIDATE THILINKAGE THIydiv (THIDATE t0,
                             const int *numdiv,
                             const int *months,
                             const THIDATE *anchor,
                             const int *endmnth,
                             int *status);

```

Corresponding @*nalyst*® spreadsheet function: **ydiv**

Returns the date of the beginning of the year division (e.g. the beginning of the current quarter) in which given date falls, or that is a given number of divisions away.

Parameter	Description	Example
t0	given date	THIdate (97,6,15,0)
<i>numdiv</i>	pointer to number of divisions away from beginning of the division in which t0 falls (can be negative); if the pointer is NULL the current division is used	int nd = 20; <i>numdiv</i> = &nd;
<i>months</i>	pointer to number of months per division e.g. 3 = quarter, 6 = half-year; does not have to be a divisor of 12, e.g. 5; can be longer than 1 year, e.g. 18; if the pointer is NULL the default value of 3 (quarters) is used	int mn = 6; <i>months</i> = &mn;
<i>anchor</i>	pointer to date to anchor division boundaries on; if the pointer is NULL the default date of 1-Jan-1900 is used	* <i>anchor</i> = THI-date (95,4,15,0);
<i>endmnth</i>	pointer to indicate forced adherence to ends of months 1 = on and 0 = off; if the pointer is NULL the default is adherence to the end of the month	int eom = 1; <i>endmnth</i> = &eom;
<i>status</i>	optional returned status value (pass NULL if error checking is not desired)—nonzero indicates an error	int istatus; <i>status</i> = &istatus;

Required Header Files:

- THIdate.h

Differences From Spreadsheet Version:

- The *status* variable may be supplied to indicate the return status.

```
double THILINKAGE THIbyld (const double *coupon,
                           int num_coupon,
                           const THIDATE *maturity,
                           int num_maturity,
                           THIDATE settle,
                           double price,
                           const THIDATE *dated,
                           const THIDATE *firstcoup,
                           const int *bondtype,
                           const int *freq,
                           int num_freq,
                           const double *redem,
                           const double *itax,
                           const double *gtax,
                           const int *cutoff,
                           const double *issue_pr,
                           const int *true_yld,
                           const int *wkend,
                           const THIDATE *holidays,
                           int numholidays,
                           const double *rout,
                           const THIDATE *rstart,
                           int num_rstart,
                           const THIDATE *rend,
                           int num_rend,
                           const double *ramount,
                           const double *voluntary,
                           const int *numexdiv,
                           int *status);
```

Corresponding @*nalyst*[®] spreadsheet function: **byld**

Calculates the yield-to-maturity (or yield-to-call) for a bond given price.

Parameter	Description	Example
coupon	pointer to single coupon rate, or an array of rates for stepped coupon bonds)	coupon [] = {0.084};
num_coupon	size of the coupon array	1
maturity	pointer to maturity date, or an array of coupon conversion dates for stepped coupon bonds (with the last date being the maturity date)	* maturity = THI-date (105,6,30,0);
num_maturity	size of the maturity array	1
settle	settlement date	THIdate (98,6,16,0)
price	quoted price (par 100, not including accrued interest)	99.00
<i>dated</i>	pointer to dated date; NULL indicates none specified	* <i>dated</i> = THI-date (96,6,1,0);

Parameter	Description	Example
<i>firstcoup</i>	pointer to first coupon date; NULL indicates none specified	<code>*firstcoup = THI-date(97,5,31,0);</code>
<i>bondtype</i>	pointer to bond type code; NULL indicates a U.S. Treasury bond	<code>*bondtype = THI-BOND_AGENCY;</code>
<i>freq</i>	pointer to frequency of coupon payments; or two-element array with the first element the coupon frequency and the second element the redemption frequency (for sinking fund bonds); if NULL the frequency is inferred from <i>bondtype</i>	<code>freq[] = {4};</code>
<i>num_freq</i>	the number of elements in the <i>freq</i> array (should be 0, 1, or 2)	1
<i>redem</i>	pointer to redemption value; if NULL the default of 100.00 is used	<code>*redem = 101.50;</code>
<i>itax</i>	pointer to marginal tax rate for ordinary income; if NULL there is no tax	<code>*itax = 0.31;</code>
<i>gtax</i>	pointer to marginal tax rate for long-term capital gains; if NULL there is no tax	<code>*itax = 0.28;</code>
<i>cutoff</i>	pointer to holding period term: if $*cutoff \leq 24$, then months from <i>settle</i> ; if $*cutoff > 24$, then days from <i>settle</i> ; if NULL the default of 12 months from <i>settle</i> is used	<code>*cutoff = 6;</code>
<i>issue_pr</i>	pointer to issue price; if NULL the default of 100.00 is used	<code>*issue_pr = 98.75;</code>
<i>true_yld</i>	pointer to flag specifying calculation method: NULL/0 = default calculation method (net yield), 1 = true yield, -1 = ISMA, -2 = simple (Japanese), -3 = Braeß-Fangmeyer, -4 = Moosmüller, -5 = consortium, -6 = force act/act discounting	<code>int ty = 1; true_yld = &ty;</code>
<i>wkend</i>	pointer to flag specifying whether Saturday and/or Sunday are business days— NULL/0: neither is a business day; 1: Saturday is a business day; 2: Sunday is a business day; 3: both are business days	<code>int wk = 1; wkend = &wk;</code>
<i>holidays</i>	array of date serial numbers (in increasing order) that are holidays; NULL indicates no holidays	<code>holidays[] = {...};</code>
<i>numholidays</i>	the size of the <i>holidays</i> array	4
<i>rou</i>	pointer to flag to indicate the type of redemption schedule for a sinking fund bond: NULL/0: no early redemption -1: = uniform redemption schedule 1: custom redemption schedule; <i>r</i> = a fraction between 0 and 1 equal to the percent of debt outstanding at settlement (for custom schedule with voluntary payments)	<code>*rou = 0.5;</code>
<i>rstart</i>	pointer to a single start date (for use with a uniform redemption schedule) or an array of start dates (for use with a custom redemption schedule)	<code>rstart[] = {...};</code>
<i>num_rstart</i>	the size of the <i>rstart</i> array	5

Parameter	Description	Example
<i>rend</i>	optional array of end dates for custom redemption schedules	<i>rend</i> [] = {···};
<i>num_rend</i>	the size of the <i>rend</i> array	5
<i>ramount</i>	array of sinking amounts expressed as fractions (between 0 and 1) equal to [redemption payment/debt issued]—for use with custom redemption (should be NULL for other than custom redemption)	<i>ramount</i> [] = {···};
<i>voluntary</i>	pointer to the ratio of voluntary payments to mandatory payments; if NULL there are no voluntary payments	* <i>voluntary</i> = 1.5;
<i>numexdiv</i>	pointer to the number of days the bond trades ex-dividend prior to the coupon date. Use negative integers to specify calendar days; positive integers to specify business days; if NULL the value is inferred from <i>bondtype</i>	* <i>numexdiv</i> = 3;
<i>status</i>	returned status value—nonzero indicates an error; NULL signifies no error checking	int <i>istatus</i> ; <i>status</i> = & <i>istatus</i> ;

Required Header Files:

- THlbonds.h

Differences From Spreadsheet Version:

- The parameters *num_coupon*, *num_maturity*, *num_freq*, *num_holidays*, *num_rstart* and *num_rend* must be supplied to indicate the sizes of the respective arrays.
- The holidays must be given in chronological order.
- The *status* variable may be supplied to indicate the return status.