date

COLLABORATORS			
	TITLE : date		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		April 15, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

date

1.1 date.doc

```
--background--

Compare2Dates()

Compare2Times()

DateText()

DateToJD()

DateToNum()

DayDiff()

DaysAfterWeekday()

DaysBeforeWeekday()

DiffDateRange()

DiffTime()

Easter()

FormatDate()

FormatTime()

GMTToLocal()

GregorianDayDiff()

GregorianDaysAfterWeekday()

GregorianDaysBeforeWeekday()

GregorianDiffDate()
```

GregorianDiffDateRange()

GregorianEaster()

GregorianLastMonthDay()

GregorianLeapYear()

GregorianMonthDays()

GregorianMoonAge()

GregorianMoonPhase()

GregorianRangeDiff()

GregorianToJD()

GregorianWeek()

GregorianWeekday()

GregorianWWtoDM()

GregorianYearDays()

GSYearToJD()

GYearToScaliger()

HeisDayDiff()

HeisDaysAfterWeekday()

HeisDaysBeforeWeekday()

HeisDiffDate()

HeisDiffDateRange()

HeisEaster()

HeisLastMonthDay()

HeisLeapYear()

HeisMonthDays()

HeisRangeDiff()

HeisToJD()

HeisWeek()

HeisWeekday()

HeisWWtoDM()

HeisYearDays()

HSYearToJD()

HYearToScaliger()

JDToDate()

JDToGregorian()

JDToHeis()

JDToJulian()

JDtoMJD()

JDToTime()

JSYearToJD()

JulianDayDiff()

JulianDaysAfterWeekday()

JulianDaysBeforeWeekday()

JulianDiffDate()

JulianDiffDateRange()

JulianEaster()

JulianLastMonthDay()

JulianLeapYear()

JulianMonthDays()

JulianRangeDiff()

JulianToJD()

JulianWeek()

JulianWeekday()

JulianWWtoDM()

JulianYearDays()

JYearToScaliger()

LastMonthDay()

LeapYear()

LMT

LocalToGMT()

MJDtoJD()

MonthDays()

MonthShortText()

MonthText()

NextValidDate()

NextValidGregorianDate()

NextValidHeisDate()

NextValidJulianDate()

NumToDate()

NumToTime()

ParseDate()

ParseTime()

PreviousValidDate()

PreviousValidGregorianDate()

PreviousValidHeisDate()

PreviousValidJulianDate()

RangeDiff()

ScaligerYearToG()

ScaligerYearToH()

ScaligerYearToJ()

ScaligerYearToYear()

SecToTime()

SetCountry()

SetFirstWeekday()

SupplementCentury()

SYearToJD()

1.2 Date/--background--

```
 NAME
Date -- This module was designed in helping calc.calendar dates (V33)

 FUNCTION
This module has been designed to become a useful and portable library
and to help developers calculate calendar dates!

 NOTES
A tropical year is 365.2422 days! / 365d, 5h, 48min, 46sec
A moon month is 29.53059 days! / 29d, 12h, 44min, 2.9 sec
A moon phase is 7.38265 days!
The calculations are historical and NOT astronomical!
```

1.3 Date/Compare2Dates

```
                NAME
Compare2Dates -- Compares date1 with date2. (V33.100)

 SYNOPSIS
compare = date_Compare2Dates(day1,month1,year1,day2,month2,year2);
  d0            d0    d1    d2  d3    d4    d5

short date_Compare2Dates(const unsigned short day1,
    const unsigned short month1, const long year1,
    const unsigned short day2, const unsigned short month2,
    const long year2);

 FUNCTION
Compare2Dates compares date1 with date2.

 INPUTS
day1   - day of the first date
month1 - month of the first date
year1  - year of the first date
day2   - day of the second date
month2 - month of the second month
year2  - year of the second date

 RESULT
compare - -1 : date1 < date2
     0 : date1 = date2
     1 : date1 > date2

 EXAMPLE
...
if (date_Compare2Dates(18,9,1970,22,1,1994) == -1)
 {
  printf("<\n");
 }
else
 {
  printf(">=\n");
 }
...

 NOTES
It is better only to use this function for years from 8 to 8000!
There is no need for different versions for Julian, Gregorian and
Heis dates!

 BUGS
There is no check if the dates are valid!

 SEE ALSO

              Compare2Times()
```

1.4 Date/Compare2Times

```
                  NAME
Compare2Times -- Compares time1 with time2. (V33.100)

 SYNOPSIS
compare = time_Compare2Times(hour1,min1,sec1,hour2,min2,sec2);
  d0            d0    d1   d2   d3    d4   d5

short time_Compare2Times(const unsigned short hour1,
    const unsigned short min1, const unsigned short sec1,
    const unsigned short hour2, const unsigned short min2,
    const unsigned short sec2);

 FUNCTION
Compare2Times compares time1 with time2 (24h format only).

 INPUTS
hour1 - Hour of the first time.
min1  - Minute of the first time.
sec1  - Second of the first time.
hour2 - Hour of the second time.
min2  - Minute of the second time.
sec2  - Second of the second time.

 RESULT
compare - -1 : time1 < time2
     0 : time1 = time2
     1 : time1 > time2

 EXAMPLE
...
if (time_Compare2Times(13,10,0,9,0,0) == -1)
  printf("<\n");
else
  printf(">=\n");
...

 NOTES
This compares two times of 24h format!

 BUGS
There is no check if the times are valid times!

 SEE ALSO

              Compare2Dates()
```

1.5 Date/DateText

```
                  NAME
DateText -- Get a date text string. (V33.130)
```

```
 SYNOPSIS
date_DateText(dt,text,lang,wc);
        d0  a0   d1  d2

void date_DateText(const date_DateTexts dt, char *const text,
    const date_Languages lang, const date_WordClass wc);

 FUNCTION
This function gets a date specified text in each supported language

 INPUTS
dt     - The text you want.
text   - Pointer to a string to fill in the text.
lang   - Language for which you want the text.
wc     - singular   : The singular form is given.
   plural     : The plural form is given (if available).
   periodical : The periodical form is given (if available).

 RESULT
None

 EXAMPLE
...
char txt[20];
...
date_DateText(day,&txt,English,date_SINGULAR);
...

 NOTES
Languages:
Locale    : This is an Amiga >= OS2.1 only feature, for <= OS2.0
      and other systems it will return English text!

Available strings:
day,month,year,week,weekday,hour,minute,second,yesterday,today,
tomorrow

There is no plural form available for: yesterday,today,tomorrow
There is no periodical form available for: yesterday,today,tomorrow,
weekday

 BUGS
In this version there is no check, if there is enough space in
text!

 SEE ALSO

             WeekdayText()
             ,
             WeekdayShortText()
             ,
             MonthText()
             ,
             MonthShortText()
```

1.6 Date/DateToJD

```
                  NAME
DateToJD -- Returns the JD for a date. (V33.310)

 SYNOPSIS
jd = date_DateToJD(day,month,year,calendar);
d0             d0   d1    d2    d3

unsigned long date_DateToJD(const unsigned short day,
    const unsigned short month, const long year,
    const date_Calendars calendar);

 FUNCTION
Returns the JD for a  date.

 INPUTS
day  - day of the date to convert
month  - month of the date to convert
year   - year of the date to convert
calendar - Calendar system to use

 RESULT
jd - This is the JD

 EXAMPLE
...
jd = date_DateToJD(23,1,1994,date_Gregorian);
...

 NOTES
It is better to use this function only from 8 to 8000!

 BUGS
unknown.

 SEE ALSO

              JulianToJD()
              ,
              GregorianToJD()
              ,
              HeisToJD()
```

1.7 Date/DateToNum

```
                  NAME
DateToNum -- Returns the date in a numeric format. (V33.250)

 SYNOPSIS
num = date_DateToNum(day,month,year);
d0         d0   d1  d2
```

```
long date_DateToNum(const unsigned short day,
    const unsigned short month, const long year);

 FUNCTION
Returns the date in a numeric format:
(year*100+month)*100+day

 INPUTS
day  - day of the date to convert
month  - month of the date to convert
year   - year of the date to convert

 RESULT
num - Date in numeric format.

 EXAMPLE
...
num = date_DateToNum(28,6,1997);
...

 NOTES
Negative years will be handled correctly.

 BUGS
None.

 SEE ALSO

              NumToDate()
```

1.8 Date/DayDiff

```
                 NAME
DayDiff -- Calculates the days between 2 dates. (V33.310)

 SYNOPSIS
days = date_DayDiff(day1,month1,year1,day2,month2,year2,calendar);
 d0        d0    d1   d2  d3    d4    d5    d6

long date_DayDiff(const unsigned short day1,
    unsigned short month1, long year1, const unsigned short day2,
    unsigned short month2, long year2, const date_Calendars calendar);

 FUNCTION
DayDiff gives you back the number of days between
two specified dates.

 INPUTS
day1     - day of the first date
month1   - month of the first date
year1    - year of the first date
day2     - day of the second date
month2   - month of the second month
year2    - year of the second date
```

```
calendar - Calendar system to use

 RESULT
days - The number of days between the two dates
    (positive if date1 <= date2).

 EXAMPLE
long days;
...
days = date_DayDiff(18,9,1970,22,1,1994,date_Gregorian);
printf("Age of Kai Hofmann in days : %ld\n",days);
...

 NOTES
It is better only to use this function for years from 8 to 02.3200!

 BUGS
If you use one of the dates 5.10.1582 to 14.10.1582 you will get an
incorrect output because these days don't exist!

 SEE ALSO

              JulianDayDiff()
              ,
              GregorianDayDiff()
              ,
              HeisDayDiff()
```

1.9 Date/DaysAfterWeekday

```
                  NAME
DaysAfterWeekday -- Returns the diff to wday after. (V33.310)

 SYNOPSIS
days = date_DaysAfterWeekday(day,month,year,weekday,calendar);
 d0          d0   d1   d2 d3   d4

unsigned short date_DaysAfterWeekday(const unsigned short day,
    const unsigned short month, const long year,
    const date_Weekdays weekday, const date_Calendars calendar);

 FUNCTION
Returns the days to the weekday after the specified date.

 INPUTS
day  - day of the date
month  - month of the date
year   - year of the date
weekday  - weekday to search for building difference
calendar - Calendar system to use

 RESULT
days - The days after to the searched weekday.
```

```
 EXAMPLE
...
days = date_DaysAfterWeekday(22,1,1994,date_Thursday,date_Gregorian);
...

 NOTES
It is better to use this function only from 8 to 3200!

 BUGS
See Weekday()!

 SEE ALSO

              JulianDaysAfterWeekday()
              ,
              GregorianDaysAfterWeekday()
              ,

              HeisDaysAfterWeekday()
```

1.10 Date/DaysBeforeWeekday

```
                  NAME
DaysBeforeWeekday -- Returns the diff to wday before. (V33.310)

 SYNOPSIS
days = date_DaysBeforeWeekday(day,month,year,weekday,calendar);
 d0            d0  d1   d2  d3     d4

unsigned short date_DaysBeforeWeekday(const unsigned short day,
    const unsigned short month, const long year,
    const date_Weekdays weekday, const date_Calendars calendar);

 FUNCTION
Returns the days to the weekday before the specified date.

 INPUTS
day  - day of the date
month  - month of the date
year   - year of the date
weekday  - weekday to search for building difference
calendar - Calendar system to use

 RESULT
days - The days gets you back to the searched weekday (1-7)
    If you get back an 8, an error occurs!

 EXAMPLE
...
days = date_DaysBeforeWeekday(22,1,1994,date_Thursday,date_Gregorian);
...

 NOTES
It is better to use this function only from 8 to 3200!
```

```
 BUGS
See Weekday()!

 SEE ALSO

              JulianDaysBeforeWeekday()
              ,
              GregorianDaysBeforeWeekday()
              ,

              HeisDaysBeforeWeekday()
```

1.11 Date/DiffDateRange

```
                  NAME
DiffDateRange -- Calc new date from old one. (V33.310)

 SYNOPSIS
date_DiffDateRange(day,month,year,days,months,
    years,dday,dmonth,dyear,calendar);
        d0   d1   d2   d3   d4
     d5    a0  a1    a2     d6

void date_DiffDateRange(const unsigned short day,
    const unsigned short month, const long year, long days,
    long months, long years, unsigned short *const dday,
    unsigned short *const dmonth, long *const dyear,
    const date_Calendars calendar);

void date_DiffDateRange(const unsigned short day,
    const unsigned short month, const long year, long days,
    long months, long years, unsigned short &dday,
    unsigned short &dmonth, long &dyear,
    const date_Calendars calendar);

 FUNCTION
Returns the date which lies diffrange before/after the specified date.

 INPUTS
day      - day of the date
month    - month of the date
year     - year of the date
days     - difference to the date in days
months   - difference to the date in months
years    - difference to the date in years
calendar - Calendar system to use

 RESULT
dday   - Destination day
dmonth - Destination month
dyear  - Destination year

 EXAMPLE
```

```
...
date_DiffDateRange(23,1,1994,0,0,1,&dday,&dmonth,&dyear,date_Gregorian);
...

 NOTES
It is better to use this function only from 8 to 3200!

 BUGS
unknown.

 SEE ALSO

              RangeDiff()
              ,
              JulianDiffDateRange()
              ,
              GregorianDiffDateRange()
              ,

              HeisDiffDateRange()
```

1.12 Date/DiffTime

```
                  NAME
DiffTime -- Returns the diff. time to another time. (V33)

 SYNOPSIS
time_DiffTime(hour,min,sec,secs,rhour,rmin,rsec);
         d0  d1  d2  d3 a0    a1   a2

void time_DiffTime(const unsigned short hour, const unsigned short min,
    const unsigned short sec, long secs, unsigned short *const rhour,
    unsigned short *const rmin, unsigned short *const rsec);

void time_DiffTime(const unsigned short hour, const unsigned short min,
    const unsigned short sec, long secs, unsigned short &rhour,
    unsigned short &rmin, unsigned short &rsec);

 FUNCTION
Returns the time which lies diffsecs before/after the specified time.

 INPUTS
hour   - hour
min  - minute
sec  - second
diffsecs - difference to the time in seconds

 RESULT
rhour - new hour
rmin  - new minute
rsec  - new second

 EXAMPLE
...
```

```
time_DiffTime(12,19,0,2460,&hour,&min,&sec);
...

 NOTES
Don't forget to convert AM/PM to 24h time!
Don't forget to convert 24h time to AM/PM time if needed!

 BUGS
No errorcheck, if you use a valid time

 SEE ALSO

              TimeToSec()
              ,
              SecToTime()
```

1.13 Date/Easter

```
                 NAME
Easter -- Returns the date of Easter in a year (V33.310)

 SYNOPSIS
date_Easter(year,dday,dmonth,calendar);
        d0   a0  a1    d1

void date_Easter(const long year, unsigned short *const dday,
    unsigned short *const dmonth, const date_Calendars calendar);

void date_Easter(const long year, unsigned short &dday,
    unsigned short &dmonth, const date_Calendars calendar);

 FUNCTION
Returns the date of Easter for a specified year.

 INPUTS
year     - Easter is calculated for this year
calendar - Calendar system to use

 RESULT
dday   - day of Easter Sunday
dmonth - month of Easter Sunday

 EXAMPLE
...
date_Easter(1994,&dday,&dmonth,date_Gregorian);
...

 NOTES
Use this only for 31 to 2099!

 BUGS
None.

 SEE ALSO
```

```
JulianEaster()
,
GregorianEaster()
,
HeisEaster()
```

1.14 Date/FormatDate

```
                NAME
FormatDate -- Formats a date string (V33.238)

 SYNOPSIS
date_FormatDate(fmt,day,month,year,lang,tstr);
     a0  d0  d1    d2   d3   a1

void date_FormatDate(const char *const fmt,
   const unsigned short day, const unsigned short month,
   const long year, const date_Languages lang, char *const tstr);

 FUNCTION
Formats the date into a string.

 INPUTS
fmt   - Format template string
day   - Day of the date to format
month   - Month of the date to format
year    - Year of the date to format
lang    - Language to use for strings

 RESULT
tstr  - Formatted date string

 EXAMPLE
char buffer[20];
...
date_FormatDate("%Y-%m-%d",10,3,1997,date_Locale,buffer);
/* "1997-03-10" */
...

 SYNTAX
Syntax of Amiga compatible % commands:
%d : Day number with leading 0s
%e : Day number with leading spaces
%m : Month number with leading 0s
%h : Abbreviated month name
%b : Abbreviated month name
%B : Month name
%y : Year using two digits with leading 0s
%Y : Year using four digits with leading 0s
%j : Julian date
%w : Weekday number
%a : Abbreviated weekday name
%A : Weekday name
```

```
%U : Week number, taking Sunday as first day of week
%W : Week number, taking Monday as first day of week
%x : Same as "%m/%d/%y"
%D : Same as "%m/%d/%y"

Syntax of % commands:
%Ddf  : Day with leading 0s
%Ddv  : Day without leading 0s
%DDf  : Day within the year with leading 0s
%DDv  : Day within the year without leading 0s
%Dmf  : Month with leading 0s
%Dmv  : Month without leading 0s
%Dms  : Month string
%Dma  : Abbreviated month string
%Dy2f : 2-digit year with leading 0s
%Dy2v : 2-digit year without leading 0s
%Dy4f : 4-digit year with leading 0s
%Dy4v : 4-digit year without leading 0s
%Dys  : Scaliger year
%Dj   : JD date
%DJ   : MJD date
%Dwn  : Weekday number (1-7)
%Dws  : Weekday string
%Dwa  : Abbreviated weekday string
%DWf  : Weeknumber with leading 0s
%DWv  : Weeknumber without leading 0s
%DMf  : Age of the moon (0-30 ?) with leading 0s
%DMv  : Age of the moon (0-30 ?) without leading 0s

 NOTES
None.

 BUGS
No errorcheck, if you use a valid date, no check if tstr is big
enough.
Using something like "%a %W %Y" as fmt might result in a wrong
output year if used for the first/last week of a year.
Example:
1996-12-30 results into "Mo 1 1996" instead of 1997!

 SEE ALSO

              ParseDate()
```

1.15 Date/FormatTime

```
                 NAME
FormatTime -- Formats a time string (V33.224)

 SYNOPSIS
time_FormatTime(fmt,ChangeDay,ChangeHour,DST,hour,min,sec,
   zonemin,tstr);
     a0 d0     d1  d2   d3   d4  d5
     d6     a1
```

```
void time_FormatTime(const char *const fmt,
   const enum time_ChangeDay ChangeDay,
   const unsigned short ChangeHour, const bool DST,
   const unsigned short hour,  const unsigned short min,
   const unsigned short sec,  const short zonemin,
   char *const tstr);

 FUNCTION
Formats the time into a string.

 INPUTS
fmt   - Format template string
ChangeDay - Normal day, winter to summer or summer to winter
      time change day.
ChangeHour  - Hour of summer/winter time change
DST   - Daylight saving status
hour    - Hour of the time to format
min   - Minute of the time to format
sec   - Second of the time to format
zonemin   - Time that was added to GMT time to get the local
      time zone (in minutes -779 to +779)

 RESULT
tstr  - Formatted time string

 EXAMPLE
char buffer[20];
...
time_FormatTime("%H:%M:%S",time_Normal,2,0,14,57,0,-360,
buffer); /* "14:57:0" */
...

 SYNTAX
Syntax of Amiga compatible % commands:
%q : Hour using 24-hour style
%H : Hour using 24-hour style with leading 0s
%Q : Hour using 12-hour style
%I : Hour using 12-hour style with leading 0s
%p : AM or PM strings
%M : The number of minutes with leading 0s
%S : Number of seconds with leadings 0s
%R : Same as "%H:%M"
%X : Same as "%H:%M:%S"
%T : Same as "%H:%M:%S"
%r : Same as "%I:%M:%S %p"

Syntax of % commands:
%Th1f  : 12 with leading 0s
%Th1v  : 12 without leading 0s
%Th2f  : 24 with leading 0s
%Th2v  : 24 without leading 0s
%Tpso  : a/p
%Tpsu  : A/P
%Tplo  : am/pm
%Tplu  : AM/PM
%Tmf   : with leading 0s
```

```
%Tmv   : without leading 0s
%Tsf   : with leading 0s
%Tsv   : without leading 0s
%Tj.   : starting with '.'
%Tj,   : starting with ','
%Tj0   : starting with '0.'
%Tj1   : starting with '0,'

%Tzh?? : hours only
%Tzm?  : 0100
%TzM?? : 01:00
%Tz?z? : Use Z   for UTC
%Tz?0? : Use +00 for UTC
%Tz??f : use leading 0s
%Tz??v : do not use leading 0s

%Tc1   : Use DST for s->w switch
%Tc2   : Use I/II for s->w switch
%Tc3   : Use a/b for s->w switch (24h only) - on 12h falls back to 2

 NOTES
None.

 BUGS
No errorcheck, if you use a valid time, no check if tstr is big
enough.

 SEE ALSO

              ParseTime()
```

1.16 Date/GMTToLocal

```
                 NAME
GMTToLocal -- Converts a GMT time to a locale one (V33.300)

 SYNOPSIS
datetime_GMTToLocal(gjd,gsecs,zonemin,ChangePrevDay,ChangeDay,
    ChangeNextDay,ChangeHour,ljd,lsecs,DST);
         d0   d1  d2     d3        d4
    d5       d6      a0    a1  a2

void datetime_GMTToLocal(const unsigned long gjd,
    const unsigned long gsecs, const short zonemin,
    const time_ChangeDay ChangePrevDay,
    const time_ChangeDay ChangeDay,
    const time_ChangeDay ChangeNextDay,
    const unsigned short ChangeHour, unsigned long *const ljd,
    unsigned long *const lsecs, bool *const DST);

void datetime_GMTToLocal(const unsigned long gjd,
    const unsigned long gsecs, const short zonemin,
    const time_ChangeDay ChangePrevDay,
    const time_ChangeDay ChangeDay,
```

```
    const time_ChangeDay ChangeNextDay,
    const unsigned short ChangeHour, unsigned long &ljd,
    unsigned long &lsecs, bool &DST);

 FUNCTION
Converts a GMT date/time pair into a local date/time pair.
The conversion considers the local daylight savings time as well as
the time change from winter to summer and vice versa (if any).

 INPUTS
gjd       - GMT Julian Date
gsecs       - GMT time in seconds
zonemin       - Time that was added to GMT time to get the local time
    zone (in minutes -779 to +779)
ChangePrevDay - Normal day, winter to summer or summer to winter time
    change on the previous day
ChangeDay     - Normal day, winter to summer or summer to winter time
    change day
ChangeNextDay - Normal day, winter to summer or summer to winter time
    change on the next day
ChangeHour    - Hour of summer/winter time change
DST       - Daylight saving status

 RESULT
DST   - Daylight saving status
ljd   - Local Julian Date
lsecs - Local time in seconds

 EXAMPLE
unsigned long ljd,lsecs;
bool DST = true;
...
datetime_GMTToLocal(2450919,30300,+60,time_Normal,time_Normal,
    time_Normal,2,&ljd,&lsecs,&DST);
...

 NOTES
None.

 BUGS
No errorcheck, if you use a valid date/time.

 SEE ALSO

              LocalToGMT()
              ,
              JulianToJD()
              ,
              GregorianToJD()
              ,
              HeisToJD()
              ,

              JDToJulian()
              ,
              JDToGregorian()
              ,
```

```
JDToHeis()
,
TimeToSec()
,
SecToTime()
,

TimeZoneFactor()
```

1.17 Date/GregorianDayDiff

```
                  NAME
GregorianDayDiff -- Calculates the days between 2 dates. (V33)

 SYNOPSIS
days = date_GregorianDayDiff(day1,month1,year1,day2,month2,year2);
 d0           d0    d1    d2  d3    d4    d5

long date_GregorianDayDiff(const unsigned short day1,
    unsigned short month1, long year1, const unsigned short day2,
    unsigned short month2, long year2);

 FUNCTION
GregorianDayDiff gives you back the number of days between
two specified dates.

 INPUTS
day1   - day of the first date
month1 - month of the first date
year1  - year of the first date
day2   - day of the second date
month2 - month of the second month
year2  - year of the second date

 RESULT
days - The number of days between the two dates
    (positive if date1 <= date2).

 EXAMPLE
long days;
...
days = date_GregorianDayDiff(18,9,1970,22,1,1994);
printf("Age of Kai Hofmann in days : %ld\n",days);
...

 NOTES
It is better only to use this function for years from 8 to 02.3200!

 BUGS
If you use one of the dates 5.10.1582 to 14.10.1582 you will get an
incorrect output because these days don't exist!

 SEE ALSO
```

```
               GregorianLeapYear()
               ,
               GregorianMonthDays()
               ,
               GregorianYearDays()
               ,

               JulianDayDiff()
               ,
               HeisDayDiff()
```

1.18 Date/GregorianDaysAfterWeekday

```
                  NAME
GregorianDaysAfterWeekday -- Returns the diff to wday after. (V33)

 SYNOPSIS
days = date_GregorianDaysAfterWeekday(day,month,year,weekday);
 d0              d0   d1   d2 d3

unsigned short date_GregorianDaysAfterWeekday(const unsigned short day,
    const unsigned short month, const long year,
    const date_Weekdays weekday);

 FUNCTION
Returns the days to the weekday after the specified date.
If you specify the 22.1.1994 (Saturday) and Thursday
you get back 5!
If you specify the 22.1.1994 and Saturday you get back 0
(the same day)!

 INPUTS
day - day of the date
month - month of the date
year  - year of the date
weekday - weekday to search for building difference

 RESULT
days - The days after to the searched weekday.

 EXAMPLE
...
days = date_GregorianDaysAfterWeekday(22,1,1994,Thursday);
...

 NOTES
It is better to use this function only from 8 to 3200!

 BUGS
See GregorianWeekday()!

 SEE ALSO

               GregorianWeekday()
```

```
              ,
              JulianDaysAfterWeekday()
              ,
              HeisDaysAfterWeekday()
```

1.19 Date/GregorianDaysBeforeWeekday

```
                  NAME
GregorianDaysBeforeWeekday -- Returns the diff to wday before. (V33)

 SYNOPSIS
days = date_GregorianDaysBeforeWeekday(day,month,year,weekday);
 d0         d0  d1    d2  d3

unsigned short date_GregorianDaysBeforeWeekday(const unsigned short day,
    const unsigned short month, const long year,
    const date_Weekdays weekday);

 FUNCTION
Returns the days to the weekday before the specified date.
If you specify the 22.1.1994 (Saturday) and Thursday
you get back 2!
If you specify the 22.1.1994 and Saturday you get back 0
(the same day)!

 INPUTS
day - day of the date
month - month of the date
year  - year of the date
weekday - weekday to search for building difference

 RESULT
days - The days gets you back to the searched weekday (1-7)
    If you get back an 8 an error occurs!

 EXAMPLE
...
days = date_GregorianDaysBeforeWeekday(22,1,1994,Thursday);
...

 NOTES
It is better to use this function only from 8 to 3200!

 BUGS
See GregorianWeekday()!

 SEE ALSO

              GregorianWeekday()
              ,
              JulianDaysBeforeWeekday()
              ,
              HeisDaysBeforeWeekday()
```

1.20 Date/GregorianDiffDate

```
                  NAME
GregorianDiffDate -- Returns the diff date to another date. (V33)

 SYNOPSIS
date_GregorianDiffDate(day,month,year,days,dday,dmonth,dyear);
      d0   d1   d2   d3   a0    a1  a2

void date_GregorianDiffDate(const unsigned short day,
    const unsigned short month, const long year, long days,
    unsigned short *const dday, unsigned short *const dmonth,
    long *const dyear);

void date_GregorianDiffDate(const unsigned short day,
    const unsigned short month, const long year, long days,
    unsigned short &dday, unsigned short &dmonth, long &dyear);

 FUNCTION
Returns the date which lies days before/after the specified date.

 INPUTS
day   - day of the date
month - month of the date
year  - year of the date
days  - difference to the date in days

 RESULT
dday   - Destination day
dmonth - Destination month
dyear  - Destination year

 EXAMPLE
...
date_GregorianDiffDate(23,1,1994,7,&dday,&dmonth,&dyear);
...

 NOTES
This function is OBSOLETE - please use
              GregorianDiffDateRange()
                instead!
It is better to use this function only from 8 to 3200!

 BUGS
unknown.

 SEE ALSO

              GregorianDayDiff()
              ,
              GregorianMonthDays()
              ,
              JulianDiffDate()
```

```
               ,

               HeisDiffDate()
```

1.21 Date/GregorianDiffDateRange

```
                  NAME
GregorianDiffDateRange -- Calc new date from old one. (V33.296)

 SYNOPSIS
date_GregorianDiffDateRange(day,month,year,days,months,
    years,dday,dmonth,dyear);
           d0   d1   d2   d3    d4
     d5    a0  a1    a2

void date_GregorianDiffDateRange(const unsigned short day,
    const unsigned short month, const long year, long days,
    long months, long years, unsigned short *const dday,
    unsigned short *const dmonth, long *const dyear);

void date_GregorianDiffDateRange(const unsigned short day,
    const unsigned short month, const long year, long days,
    long months, long years, unsigned short &dday,
    unsigned short &dmonth, long &dyear);

 FUNCTION
Returns the date which lies diffrange before/after the specified date.

 INPUTS
day    - day of the date
month  - month of the date
year   - year of the date
days   - difference to the date in days
months - difference to the date in months
years  - difference to the date in years

 RESULT
dday   - Destination day
dmonth - Destination month
dyear  - Destination year

 EXAMPLE
...
date_GregorianDiffDateRange(23,1,1994,0,0,1,&dday,&dmonth,&dyear);
...

 NOTES
It is better to use this function only from 8 to 3200!

 BUGS
unknown.

 SEE ALSO
```

```
                GregorianDiffDate()
                ,
                GregorianRangeDiff()
                ,
                JulianDiffDateRange()
                ,

                HeisDiffDateRange()
```

1.22 Date/GregorianEaster

```
                    NAME
GregorianEaster -- Returns the date of Easter in a year (V33)

 SYNOPSIS
date_GregorianEaster(year,dday,dmonth);
          d0   a0  a1

void date_GregorianEaster(const long year, unsigned short *const dday,
    unsigned short *const dmonth);

void date_GregorianEaster(const long year, unsigned short &dday,
    unsigned short &dmonth);

 FUNCTION
Returns the date of Easter for a specified year.

 INPUTS
year  - Easter is calculated for this year

 RESULT
dday   - day of Easter Sunday
dmonth - month of Easter Sunday

 EXAMPLE
...
date_GregorianEaster(1994,&dday,&dmonth);
...

 NOTES
Use this only for 31 to 2099!

 BUGS
None.

 SEE ALSO

                JulianEaster()
                ,
                HeisEaster()
```

1.23 Date/GregorianLastMonthDay

```
                  NAME
GregorianLastMonthDay -- The number of last day in a month. (V33.234)

 SYNOPSIS
day = date_GregorianLastMonthDay(month,year);
 d0         d0  d1

unsigned short date_GregorianLastMonthDay(const unsigned short month,
    const long year);

 FUNCTION
GregorianLastMonthDay returns the number of the last day for a given
month.

 INPUTS
month - The month from which you want to get the last day.
year  - The year in which the month is.

 RESULT
days - The number of the last day the month uses, or 0 if you use
    an incorrect month.

 EXAMPLE
...
day = date_GregorianLastMonthDay(1,1994); /* 31 */
...

 NOTES
Use this function only for years from 8 to 3199!

 BUGS
none.

 SEE ALSO

              GregorianMonthDays()
              ,
              JulianLastMonthDay()
              ,
              HeisLastMonthDay()
```

1.24 Date/GregorianLeapYear

```
                  NAME
GregorianLeapYear -- Checks if a year is a leap year. (V33)

 SYNOPSIS
leapyear = date_GregorianLeapYear(year);
   d0          d0

bool date_GregorianLeapYear(const long year);
```

```
 FUNCTION
GregorianLeapYear checks if a year is a leap year.
For years after 1582 all years devideable by 4 are leap years,
without years devideable by 100, but years devideable by 400
are leap years again!
For years before 1582 see
              JulianLeapYear()
              .

 INPUTS
year - The year which should be checked (from -32768 to 32767)
    I think only values from 8 to 3200 are valid, because of
    the variant that was done by Augustus!

 RESULT
leapyear - true if the year is a leap year, otherwise false.

 EXAMPLE
...
if (date_GregorianLeapYear(1994))
 {
  printf("leap year!\n");
 }
else
 {
  printf("no leap year!\n");
 }
...

 NOTES
A year is 365.2425 days long!
Use this function only for values from 8 to 3199!

 BUGS
No known bugs.

 SEE ALSO

              JulianLeapYear()
              ,
              HeisLeapYear()
```

1.25 Date/GregorianMonthDays

```
                 NAME
GregorianMonthDays -- Returns the number of days of a month. (V33)

 SYNOPSIS
days = date_GregorianMonthDays(month,year);
 d0       d0    d1

unsigned short date_GregorianMonthDays(const unsigned short month,
    const long year);
```

```
 FUNCTION
GregorianMonthDays returns the number of days a month in
a specified year has.
For the year 1582 and the month 10 there are only 21 days,
because of the Gregorian-reform 10 days are deleted from
the month (for more - look out for books about this!)

 INPUTS
month - The month from which you want to get the number of days.
year  - The year in which the month is.

 RESULT
days - The number of days the month uses, or 0 if you use
    an incorrect month.

 EXAMPLE
unsigned short days;
...
days = date_GregorianMonthDays(1,1994);
printf("Days of January 1994 : %hu\n",days);
...

 NOTES
Use this function only for years from 8 to 3199!

 BUGS
none.

 SEE ALSO

              GregorianLeapYear()
              ,
              JulianMonthDays()
              ,
              HeisMonthDays()
```

1.26 Date/GregorianMoonAge

```
                 NAME
GregorianMoonAge -- Returns the age of the moon (V33)

 SYNOPSIS
ep = date_GregorianMoonAge(day,month,year);
d0          d0   d1   d2

unsigned short date_GregorianMoonAge(const unsigned short day,
    const unsigned short month, const long year);

 FUNCTION
Returns the age of the moon on a specified date.

 INPUTS
day   - For this day the age is calculated.
```

```
month - For this month the age is calculated.
year  - For this year the age is calculated.

 RESULT
ep - The age of the moon on the specified date.

 EXAMPLE
...
ep = date_GregorianMoonAge(18,9,1994);
...

 NOTES
Use this only for 1582 to 4100!
This is only an experimental version!

 BUGS
unknown.

 SEE ALSO
MoonMonthAge(),GregorianEP(),
              GregorianMoonPhase()
```

1.27 Date/GregorianMoonPhase

```
 NAME
GregorianMoonPhase -- Searches for the next moon phase (V33.098)

 SYNOPSIS
jd = date_GregorianMoonPhase(day,month,year,phase);
d0           d0   d1  d2   d3

unsigned long date_GregorianMoonPhase(const unsigned short day,
    const unsigned short month, const long year,
    const MoonPhases phase);

 FUNCTION
Returns the next moon phase you are searching for after
a specified date.

 INPUTS
day   - Start day for the search.
month - Start month for the search.
year  - Start year for the search.
phase - The moon phase you want to know.

 RESULT
jd - The day (as JD) on which the moon phase was found.

 EXAMPLE
...
jd = date_GregorianMoonPhase(18,9,1994,FullMoon);
date_JDToGregorian(jd,&day,&month,&year);
...
```

```
 NOTES
The range of this function is unknown to me!
So use it only from 1583 to 2500.
This is only an experimental version!

 BUGS
unknown.

 SEE ALSO
MoonMonthAge()
```

1.28 Date/GregorianRangeDiff

```
                 NAME
GregorianRangeDiff -- Calculates the range between 2 dates. (V33.297)

 SYNOPSIS
date_GregorianRangeDiff(day1,month1,year1,day2,month2,year2,days,
    months,years);
       d0    d1    d2    d3  d4    d5    a0
      a1     a2

void date_GregorianRangeDiff(unsigned short day1,
    unsigned short month1, long year1, unsigned short day2,
    unsigned short month2, long year2, short *const days,
    short *const months, long *const years);

void date_GregorianRangeDiff(unsigned short day1,
    unsigned short month1, long year1, unsigned short day2,
    unsigned short month2, long year2, short &days, short &months,
    long &years);

 FUNCTION
GregorianRangeDiff gives you back the number of days, months and
years between two specified dates.

 INPUTS
day1   - day of the first date
month1 - month of the first date
year1  - year of the first date
day2   - day of the second date
month2 - month of the second month
year2  - year of the second date

 RESULT
days   - The number of days between the two dates
   (positive if date1 <= date2).
months - The number of months between the two dates
   (positive if date1 <= date2).
years  - The number of years between the two dates
   (positive if date1 <= date2).

 EXAMPLE
short days,months;
long years;
```

```
...
date_GregorianRangeDiff(18,9,1970,25,1,1998,&days,&months,&years);
printf("Age of Kai Hofmann is : %ld years, %hd months, %hd days\n",
   years,months,days);
...

 NOTES
It is better only to use this function for years from 8 to 02.3200!

 BUGS
If you use one of the dates 5.10.1582 to 14.10.1582 you will get an
incorrect output because these days don't exist!

 SEE ALSO

              GregorianDayDiff()
              ,
              GregorianDiffDateRange()
              ,
              JulianRangeDiff()
              ,

              HeisRangeDiff()
```

1.29 Date/GregorianToJD

```
                 NAME
GregorianToJD -- Returns the JD for a date. (V33)

 SYNOPSIS
jd = date_GregorianToJD(day,month,year);
d0       d0   d1    d2

unsigned long date_GregorianToJD(const unsigned short day,
    const unsigned short month, const long year);

 FUNCTION
Returns the JD for a Gregorian date.

 INPUTS
day  - day of the date to convert
month  - month of the date to convert
year   - year of the date to convert

 RESULT
jd - This is the JD

 EXAMPLE
...
jd = date_GregorianToJD(23,1,1994);
...

 NOTES
It is better to use this function only from 8 to 3200!
```

```
 BUGS
unknown.

 SEE ALSO

              GSYearToJD()
              ,
              GYearToScaliger()
              ,
              GregorianDayDiff()
              ,
              JulianToJD()
              ,

              HeisToJD()
```

1.30 Date/GregorianWeek

```
                  NAME
GregorianWeek -- Gets the weeknumber for a specified date. (V33)

 SYNOPSIS
weeknr = date_GregorianWeek(day,month,year);
  d0           d0   d1   d2

unsigned short date_GregorianWeek(const unsigned short day,
    const unsigned short month, const long year);

 FUNCTION
GregorianWeek gets the number of the week for a specified date.

 INPUTS
day   - day of the date
month - month of the date
year  - year of the date

 RESULT
week - This is the number of the week the specified date lies in.
    If the first day in a new year is a Friday, Saturday or
    Sunday, this would be the last week of the last year!
    If the 29.12. is a Monday, the 30.12. is a Monday or a Tuesday,
    the 31.12. is a Monday, Tuesday or a Wednesday this is the
    first week of the next year!

 EXAMPLE
...
weeknr = date_GregorianWeek(4,10,1582);
...

 NOTES
It is better only to use this function for years from 8 to 3200!

 BUGS
```

```
For years < 0 errors could occur.

 SEE ALSO

              JulianWeek()
              ,
              HeisWeek()
              ,
              GregorianWeekday()
              ,
              GregorianDayDiff()
              ,

              SetFirstWeekday()
```

1.31 Date/GregorianWeekday

```
                  NAME
GregorianWeekday -- Gets the weekday of a specified date. (V33)

 SYNOPSIS
weekday = date_GregorianWeekday(day,month,year);
  d0          d0   d1   d2

date_Weekdays date_GregorianWeekday(const unsigned short day,
    unsigned short month, long year);

 FUNCTION
GregorianWeekday gets the weekday for a specified date.

 INPUTS
day   - day of the date
month - month of the date
year  - year of the date

 RESULT
weekday - This result is of type:
    date_Weekdays = {date_dayerr,date_Monday,date_Tuesday,
         date_Wednesday,date_Thursday,date_Friday,
         date_Saturday,date_Sunday};
    dayerr will show you, that an error occurs!

 EXAMPLE
...
weekday = date_GregorianWeekday(22,1,1994);
if (weekday == dayerr)
 {
  ...
 }
...

 NOTES
It is better only to use this function for years from 8 to 3200!
In this version dayerr will only occur for the lost days :)
```

```
 BUGS
It's not possible to use years < 0 (for more see
              JulianWeekday()
              ).

 SEE ALSO

              JulianWeekday()
              ,
              HeisWeekday()
```

1.32 Date/GregorianWWtoDM

```
                  NAME
GregorianWWtoDM -- Convert Weekday,Week to Day,Month. (V33.243)

 SYNOPSIS
date_GregorianWWtoDM(weekday,week,year,dday,dmonth,dyear);
      d0    d1   d2 a0    a1     a2

void date_GregorianWWtoDM(const date_Weekdays weekday,
   const unsigned short week, const long year,
   unsigned short *const dday, unsigned short *const dmonth,
   long *const dyear);

void date_GregorianWWtoDM(const date_Weekdays weekday,
   const unsigned short week, const long year,
   unsigned short &dday, unsigned short &dmonth,
   long &dyear);

 FUNCTION
Converts weekday,week to day,month.

 INPUTS
weekday  - weekday of the date to convert
week   - week of the date to convert
year   - year of the date to convert

 RESULT
day   - day of the converted date
month - month of the converted date
year  - year of the converted date

 EXAMPLE
unsigned short day,month;
long year;
...
date_GregorianWWtoDM(date_Thursday,14,1997,&day,&month,&year);
/* 1997-04-03 */
...

 NOTES
It is better to use this function only from 8 to 3200!
```

```
Be careful, the resulting year might be different from the input
year!

 BUGS
unknown.

 SEE ALSO
GregorianWWtoDM(),
              HeisWWtoDM()
```

1.33 Date/GregorianYearDays

```
                  NAME
GregorianYearDays -- Gives back the number of days in a year. (V33)

 SYNOPSIS
days = date_GregorianYearDays(year);
 d0            d0

unsigned short date_GregorianYearDays(const long year);

 FUNCTION
GregorianYearDays gives you back the number of days in
a specified year.

 INPUTS
year  - The year in which to count the days.

 RESULT
days - The number of days the year uses.

 EXAMPLE
unsigned short days;
...
days = date_GregorianYearDays(1994);
printf("Days of 1994 : %hu\n",days);
...

 NOTES
It is better only to use this function for years from 8 to 3199!

 BUGS
No known bugs.

 SEE ALSO

              GregorianMonthDays()
              ,
              JulianYearDays()
              ,
              HeisYearDays()
```

1.34 Date/GSYearToJD

```
                  NAME
GSYearToJD -- Calcs the JD from a Scaliger year. (V33)

 SYNOPSIS
jd = date_GSYearToJD(syear);
d0          d0

unsigned long date_GSYearToJD(const unsigned long syear);

 FUNCTION
Returns the Julianday of a Scaliger year.

 INPUTS
syear   - Scaliger year

 RESULT
jd - The Julianday

 EXAMPLE
...
jd = date_GSYearToJD(4800);
...

 NOTES
It is better to use this function only from 4707 to 7981!

 BUGS
unknown.

 SEE ALSO

              JSYearToJD()
              ,
              HSYearToJD()
```

1.35 Date/GYearToScaliger

```
                  NAME
GYearToScaliger -- Returns the year as Scaliger year. (V33)

 SYNOPSIS
syear = date_GYearToScaliger(year);
 d0           d0

unsigned long date_GYearToScaliger(const long year);

 FUNCTION
Returns the Scaliger year.

 INPUTS
year   - Gregorian year
```

```
 RESULT
syear - The Scaliger year

 EXAMPLE
...
syear = date_GYearToScaliger(1994);
...

 NOTES
It is better to use this function only from 8 to 3200!

 BUGS
unknown.

 SEE ALSO

              JYearToScaliger()
              ,
              HYearToScaliger()
```

1.36 Date/HeisDayDiff

```
                  NAME
HeisDayDiff -- Calculates the days between 2 dates. (V33)

 SYNOPSIS
days = date_HeisDayDiff(day1,month1,year1,day2,month2,year2);
 d0      d0    d1    d2    d3  d4    d5

long date_HeisDayDiff(const unsigned short day1, unsigned short month1,
    long year1, const unsigned short day2, unsigned short month2,
    long year2);

 FUNCTION
HeisDayDiff gives you back the number of days between
two specified dates.

 INPUTS
day1   - day of the first date
month1 - month of the first date
year1  - year of the first date
day2   - day of the second date
month2 - month of the second month
year2  - year of the second date

 RESULT
days - The number of days between the two dates
    (positive if date1 <= date2).

 EXAMPLE
long days;
...
days = date_HeisDayDiff(18,9,1970,22,1,1994);
```

```
printf("Age of Kai Hofmann in days : %ld\n",days);
...

 NOTES
It is better only to use this function for years from 8 to 8000!

 BUGS
If you use on of the dates 5.10.1582 to 14.10.1582 you will get an
incorrect output because these days don't exist!

 SEE ALSO

              HeisLeapYear()
              ,
              HeisMonthDays()
              ,
              HeisYearDays()
              ,

              JulianDayDiff()
              ,
              GregorianDayDiff()
```

1.37 Date/HeisDaysAfterWeekday

```
                 NAME
HeisDaysAfterWeekday -- Returns the diff to the wday after. (V33)

 SYNOPSIS
days = date_HeisDaysAfterWeekday(day,month,year,weekday);
 d0         d0   d1   d2     d3

unsigned short date_HeisDaysAfterWeekday(const unsigned short day,
    const unsigned short month, const long year,
    const date_Weekdays weekday);

 FUNCTION
Returns the days to the weekday after the specified date.
If you specify the 22.1.1994 (Saturday) and Thursday
you get back 5!
If you specify the 22.1.1994 and Saturday you get back 0
(the same day)!

 INPUTS
day - day of the date
month - month of the date
year  - year of the date
weekday - weekday to search for building difference

 RESULT
days - The days after to the searched weekday.

 EXAMPLE
...
```

```
days = date_HeisDaysAfterWeekday(22,1,1994,Thursday);
...

 NOTES
It is better to use this function only from 8 to 8000!

 BUGS
See HeisWeekday()!

 SEE ALSO

              HeisWeekday()
              ,
              JulianDaysAfterWeekday()
              ,
              GregorianDaysAfterWeekday()
```

1.38 Date/HeisDaysBeforeWeekday

```
                  NAME
HeisDaysBeforeWeekday -- Returns the diff to wday before. (V33)

 SYNOPSIS
days = date_HeisDaysBeforeWeekday(day,month,year,weekday);
 d0          d0  d1    d2    d3

unsigned short date_HeisDaysBeforeWeekday(const unsigned short day,
    const unsigned short month, const long year,
    const date_Weekdays weekday);

 FUNCTION
Returns the days to the weekday before the specified date.
If you specify the 22.1.1994 (Saturday) and Thursday
you get back 2!
If you specify the 22.1.1994 and Saturday you get back 0
(the same day)!

 INPUTS
day - day of the date
month - month of the date
year  - year of the date
weekday - weekday to search for building difference

 RESULT
days - The days gets you back to the searched weekday (1-7)
    If you get back an 8 an error occurs!

 EXAMPLE
...
days = date_HeisDaysBeforeWeekday(22,1,1994,Thursday);
...

 NOTES
It is better to use this function only from 8 to 8000!
```

```
 BUGS
See HeisWeekday()!

 SEE ALSO

              HeisWeekday()
              ,
              JulianDaysBeforeWeekday()
              ,
              GregorianDaysBeforeWeekday()
```

1.39 Date/HeisDiffDate

```
                  NAME
HeisDiffDate -- Returns the date for a diff to another date. (V33)

 SYNOPSIS
date_HeisDiffDate(day,month,year,days,dday,dmonth,dyear);
       d0 d1   d2   d3   a0    a1    a2

void date_HeisDiffDate(const unsigned short day,
    const unsigned short month, const long year, long days,
    unsigned short *const dday, unsigned short *const dmonth,
    long *const dyear);

void date_HeisDiffDate(const unsigned short day,
    const unsigned short month, const long year, long days,
    unsigned short &dday, unsigned short &dmonth, long &dyear);

 FUNCTION
Returns the date which lies days before/after the specified date.

 INPUTS
day   - day of the date
month - month of the date
year  - year of the date
days  - difference to the date in days

 RESULT
dday   - Destination day
dmonth - Destination month
dyear  - Destination year

 EXAMPLE
...
date_HeisDiffDate(23,1,1994,7,&dday,&dmonth,&dyear);
...

 NOTES
This function is OBSOLETE - please use
              HeisDiffDateRange()
                instead!
It is better to use this function only from 8 to 8000!
```

```
 BUGS
unknown.

 SEE ALSO

              HeisDayDiff()
              ,
              HeisMonthDays()
              ,
              JulianDiffDate()
              ,
              GregorianDiffDate()
```

1.40 Date/HeisDiffDateRange

```
                  NAME
HeisDiffDateRange -- Calc new date from old one and diff. (V33.296)

 SYNOPSIS
date_HeisDiffDateRange(day,month,year,days,months,years,
    dday,dmonth,dyear);
      d0   d1   d2   d3    d4     d5
     a0    a1  a2

void date_HeisDiffDateRange(const unsigned short day,
    const unsigned short month, const long year, long days,
    long months, long years, unsigned short *const dday,
    unsigned short *const dmonth, long *const dyear);

void date_HeisDiffDateRange(const unsigned short day,
    const unsigned short month, const long year, long days,
    long months, long years, unsigned short &dday,
    unsigned short &dmonth, long &dyear);

 FUNCTION
Returns the date which lies diffrange before/after the specified date.

 INPUTS
day    - day of the date
month  - month of the date
year   - year of the date
days   - difference to the date in days
months - difference to the date in months
years  - difference to the date in years

 RESULT
dday   - Destination day
dmonth - Destination month
dyear  - Destination year

 EXAMPLE
...
date_HeisDiffDateRange(23,1,1994,0,0,1,&dday,&dmonth,&dyear);
```

```
...

 NOTES
It is better to use this function only from 8 to 8000!

 BUGS
unknown.

 SEE ALSO

              HeisDiffDate()
              ,
              HeisRangeDiff()
              ,
              JulianDiffDateRange()
              ,

              GregorianDiffDateRange()
```

1.41 Date/HeisEaster

```
                  NAME
HeisEaster -- Returns the date of Easter in a year (V33)

 SYNOPSIS
date_HeisEaster(year,dday,dmonth);
     d0   a0    a1

void date_HeisEaster(const long year, unsigned short *const dday,
    unsigned short *const dmonth);

void date_HeisEaster(const long year, unsigned short &dday,
    unsigned short &dmonth);

 FUNCTION
Returns the date of Easter for a specified year.

 INPUTS
year  - Easter is calculated for this year

 RESULT
dday   - day of Easter Sunday
dmonth - month of Easter Sunday

 EXAMPLE
...
date_HeisEaster(1994,&dday,&dmonth);
...

 NOTES
This is only a dummy to date_GregorianEaster!
Use this only for 31 to 2099!

 BUGS
```

```
Unknown.

 SEE ALSO

              JulianEaster()
              ,
              GregorianEaster()
```

1.42 Date/HeisLastMonthDay

```
                  NAME
HeisLastMonthDay -- The number of last day in a month. (V33.234)

 SYNOPSIS
day = date_HeisLastMonthDay(month,year);
 d0          d0    d1

unsigned short date_HeisLastMonthDay(const unsigned short month,
    const long year);

 FUNCTION
HeisLastMonthDay returns the number of the last day for a given
month.

 INPUTS
month - The month from which you want to get the last day.
year  - The year in which the month is.

 RESULT
day - The number of the last day the month uses, or 0 if you use
    an incorrect month.

 EXAMPLE
...
day = date_HeisLastMonthDays(1,1994); /* 31 */
...

 NOTES
Use this function only for years from 8 to 8000!

 BUGS
None

 SEE ALSO

              HeisMonthDays()
              ,
              JulianLastMonthDay()
              ,
              GregorianLastMonthDay()
```

1.43 Date/HeisLeapYear

```
                  NAME
HeisLeapYear -- Checks if a year is a leap year. (V33)

 SYNOPSIS
leapyear = date_HeisLeapYear(year);
   d0           d0

bool date_HeisLeapYear(const long year);

 FUNCTION
HeisLeapYear checks if a year is a leap year.
For years after 1582 see
              GregorianLeapYear()
              ,
The correction from N. Heis says, that all years devideable by
3200 are no longer leap years!
For years before 1582 see
              JulianLeapYear()
              .

 INPUTS
year - The year which should be checked (from -32768 to 32767)
    I think only values from 8 to 32767 are valid, because of
    the variant that was done by Augustus!

 RESULT
leapyear - true if the year is a leap year, otherwise false.

 EXAMPLE
...
if (date_HeisLeapYear(1994))
 {
  printf("leap year!\n");
 }
else
 {
  printf("no leap year!\n");
 }
...

 NOTES
A year is now 365.2421875 days!
Use this function only for values from 8 to 8000!

 BUGS
No known bugs.

 SEE ALSO

              JulianLeapYear()
              ,
              GregorianLeapYear()
```

1.44 Date/HeisMonthDays

```
                  NAME
HeisMonthDays -- Returns the number of days of a month. (V33)

 SYNOPSIS
days = date_HeisMonthDays(month,year);
 d0        d0  d1

unsigned short date_HeisMonthDays(const unsigned short month,
    const long year);

 FUNCTION
HeisMonthDays returns the number of days a month in
a specified year has.
For the year 1582 and the month 10 there are only 21 days,
because of the Gregorian-reform 10 days are deleted from
the month (for more - look out for books about this!)

 INPUTS
month - The month from which you want to get the number of days.
year  - The year in which the month is.

 RESULT
days - The number of days the month uses, or 0 if you use
    an incorrect month.

 EXAMPLE
unsigned short days;
...
days = date_HeisMonthDays(1,1994);
printf("Days of January 1994 : %hu\n",days);
...

 NOTES
Use this function only for years from 8 to 8000!

 BUGS
See GregorianMonthDays!

 SEE ALSO

              HeisLeapYear()
              ,
              JulianMonthDays()
              ,
              GregorianMonthDays()
```

1.45 Date/HeisRangeDiff

```
                  NAME
HeisRangeDiff -- Calculates the range between 2 dates. (V33.297)
```

```
 SYNOPSIS
date_HeisRangeDiff(day1,month1,year1,day2,month2,year2,days,months,
    years);
        d0    d1  d2    d3    d4    d5  a0    a1
     a2

void date_HeisRangeDiff(unsigned short day1,
    unsigned short month1, long year1, unsigned short day2,
    unsigned short month2, long year2, short *const days,
    short *const months, long *const years);

void date_HeisRangeDiff(unsigned short day1,
    unsigned short month1, long year1, unsigned short day2,
    unsigned short month2, long year2, short &days, short &months,
    long &years);

 FUNCTION
HeisRangeDiff gives you back the number of days, months and years
between two specified dates.

 INPUTS
day1   - day of the first date
month1 - month of the first date
year1  - year of the first date
day2   - day of the second date
month2 - month of the second month
year2  - year of the second date

 RESULT
days   - The number of days between the two dates
   (positive if date1 <= date2).
months - The number of months between the two dates
   (positive if date1 <= date2).
years  - The number of years between the two dates
   (positive if date1 <= date2).

 EXAMPLE
short days,months;
long years;
...
date_HeisRangeDiff(18,9,1970,25,1,1994);
printf("Age of Kai Hofmann is : %ld years, %hd months, %hd days\n",
   years,months,days);
...

 NOTES
It is better only to use this function for years from 8 to 8000!

 BUGS
If you use on of the dates 5.10.1582 to 14.10.1582 you will get an
incorrect output because these days don't exist!

 SEE ALSO

              HeisDayDiff()
              ,
              HeisDiffDateRange()
```

,

JulianRangeDiff()
,
GregorianRangeDiff()

1.46 Date/HeisToJD

```
                  NAME
HeisToJD -- Returns the JD for a date. (V33)

 SYNOPSIS
jd = date_HeisToJD(day,month,year);
d0        d0   d1    d2

unsigned long date_HeisToJD(const unsigned short day,
    const unsigned short month, const long year);

 FUNCTION
Returns the JD for a Heis date.

 INPUTS
day  - day of the date to convert
month  - month of the date to convert
year   - year of the date to convert

 RESULT
jd - This is the JD

 EXAMPLE
...
jd = date_HeisToJD(23,1,1994);
...

 NOTES
It is better to use this function only from 8 to 3267!

 BUGS
unknown.

 SEE ALSO

              HSYearToJD()
              ,
              HYearToScaliger()
              ,
              HeisDayDiff()
              ,
              JulianToJD()
              ,HeisToJD()
```

1.47 Date/HeisWeek

```
                  NAME
HeisWeek -- Gets the number of the week for a specified date. (V33)

 SYNOPSIS
weeknr = date_HeisWeek(day,month,year);
  d0      d0  d1    d2

unsigned short date_HeisWeek(const unsigned short day,
    const unsigned short month, const long year);

 FUNCTION
HeisWeek gets the number of the week for a specified date.

 INPUTS
day   - day of the date
month - month of the date
year  - year of the date

 RESULT
week - This is the number of the week the specified date lies in.
    If the first day in a new year is a Friday, Saturday or
    Sunday, this would be the last week of the last year!
    If the 29.12. is a Monday, the 30.12. is a Monday or a Tuesday,
    the 31.12. is a Monday, Tuesday or a Wednesday this is the
    first week of the next year!

 EXAMPLE
...
weeknr = date_HeisWeek(4,10,1582);
...

 NOTES
It is better only to use this function for years from 8 to 8000!

 BUGS
For years < 0 errors could occur.

 SEE ALSO

              JulianWeek()
              ,
              GregorianWeek()
              ,
              HeisWeekday()
              ,
              HeisDayDiff()
              ,

              SetFirstWeekday()
```

1.48 Date/HeisWeekday

```
                  NAME
HeisWeekday -- Gets the weekday of a specified date. (V33)

 SYNOPSIS
weekday = date_HeisWeekday(day,month,year);
  d0          d0   d1   d2

date_Weekdays date_HeisWeekday(const unsigned short day,
    unsigned short month, long year);

 FUNCTION
HeisWeekday gets the weekday for a specified date.

 INPUTS
day   - day of the date
month - month of the date
year  - year of the date

 RESULT
weekday - This result is of type:
    date_Weekdays = {date_dayerr,date_Monday,date_Tuesday,
         date_Wednesday,date_Thrusday,date_Friday,
         date_Saturday,date_Sunday};
    date_dayerr will show you, that an error occurs!

 EXAMPLE
...
weekday = date_HeisWeekday(22,1,1994);
if (weekday == date_dayerr)
 {
  ...
 }
...

 NOTES
It is better only to use this function for years from 8 to 8000!
In this version date_dayerr will only occur for the lost days :)

 BUGS
It is not possible to use year < 0 (see
              JulianWeekday()
               for more).

 SEE ALSO

              JulianWeekday()
              ,
              GregorianWeekday()
```

1.49 Date/HeisWWtoDM

```
                 NAME
HeisWWtoDM -- Convert Weekday,Week to Day,Month. (V33.243)

 SYNOPSIS
date_HeisWWtoDM(weekday,week,year,dday,dmonth,dyear);
       d0  d1   d2   a0  a1 a2

void date_HeisWWtoDM(const date_Weekdays weekday,
   const unsigned short week, const long year,
   unsigned short *const dday, unsigned short *const dmonth,
   long *const dyear);

void date_HeisWWtoDM(const date_Weekdays weekday,
   const unsigned short week, const long year,
   unsigned short &dday, unsigned short &dmonth,
   long &dyear);

 FUNCTION
Converts weekday,week to day,month.

 INPUTS
weekday  - weekday of the date to convert
week   - week of the date to convert
year   - year of the date to convert

 RESULT
day   - day of the converted date
month - month of the converted date
year  - year of the converted date

 EXAMPLE
unsigned short day,month;
long year;
...
date_HeisWWtoDM(date_Thursday,14,1997,&day,&month,&year);
/* 1997-04-03 */
...

 NOTES
It is better to use this function only from 8 to 8000!
Be careful, the resulting year might be different from the input
year!

 BUGS
unknown.

 SEE ALSO

             GregorianWWtoDM()
             , HeisWWtoDM()
```

1.50 Date/HeisYearDays

```
                  NAME
HeisYearDays -- Gives back the number of days in a year. (V33)

 SYNOPSIS
days = date_HeisYearDays(year);
 d0       d0

unsigned short date_HeisYearDays(const long year);

 FUNCTION
HeisYearDays gives you back the number of days in
a specified year.

 INPUTS
year  - The year in which to count the days.

 RESULT
days - The number of days the year uses.

 EXAMPLE
unsigned short days;
...
days = date_HeisYearDays(1994);
printf("Days of 1994 : %hu\n",days);
...

 NOTES
It is better only to use this function for years from 8 to 8000!

 BUGS
No known bugs.

 SEE ALSO

              HeisMonthDays()
              ,
              JulianYearDays()
              ,
              GregorianYearDays()
```

1.51 Date/HSYearToJD

```
                  NAME
HSYearToJD -- Calcs the JD from a Scaliger year. (V33)

 SYNOPSIS
jd = date_HSYearToJD(syear);
d0          d0

unsigned long date_HSYearToJD(const unsigned long syear);

 FUNCTION
Returns the Julianday of a Scaliger year.
```

```
 INPUTS
syear   - Scaliger year

 RESULT
jd - The Julianday

 EXAMPLE
...
jd = date_HSYearToJD(6700);
...

 NOTES
It is better to use this function only from 4707 to 7981!
In this version only date_GSYearToJD() is called, because the
Scaliger period is only valid to 3268

 BUGS
unknown.

 SEE ALSO

              JSYearToJD()
              ,
              GSYearToJD()
```

1.52 Date/HYearToScaliger

```
                  NAME
HYearToScaliger -- Returns the year as Scaliger year. (V33)

 SYNOPSIS
syear = date_HYearToScaliger(year);
 d0           d0

unsigned long date_HYearToScaliger(const long year);

 FUNCTION
Returns the Scaliger year.

 INPUTS
year   - Heis year

 RESULT
syear - The Scaliger year

 EXAMPLE
...
syear = date_HYearToScaliger(1994);
...

 NOTES
It is better to use this function only from 8 to 8000!
```

```
 BUGS
The Scaliger period is defined to 3268!!!.

 SEE ALSO

              JYearToScaliger()
              ,
              GYearToScaliger()
```

1.53 Date/JDToDate

```
                  NAME
JDToDate -- Returns the  date for a JD. (V33.310)

 SYNOPSIS
date_JDToDate(jd,day,month,year,calendar);
        d0 a0   a1    a2    d1

void date_JDToDate(const unsigned long jd, unsigned short *const day,
    unsigned short *const month, long *const year,
    const date_Calendars calendar);

void date_JDToDate(const unsigned long jd, unsigned short &day,
    unsigned short &month, long &year,
    const date_Calendars calendar);

 FUNCTION
Returns the  date for a JD.

 INPUTS
jd   - This is the given JD.
calendar - Calendar system to use

 RESULT
day - Day of the date.
month - Month of the date.
year  - Year of the date.

 EXAMPLE
...
date_JDToDate(2299161,&day,&month,&year,date_Gregorian);
...

 NOTES
It is better to use this function only from 1718867 to 2889835!

 BUGS
unknown.

 SEE ALSO

              JDToJulian()
              ,
              JDToGregorian()
```

,
JDToHeis()

1.54 Date/JDToGregorian

```
                  NAME
JDToGregorian -- Returns the Gregorian date for a JD. (V33.095)

 SYNOPSIS
date_JDToGregorian(jd,day,month,year);
       d0 a0   a1  a2

void date_JDToGregorian(const unsigned long jd, unsigned short *const day,
    unsigned short *const month, long *const year);

void date_JDToGregorian(const unsigned long jd, unsigned short &day,
    unsigned short &month, long &year);

 FUNCTION
Returns the Gregorian date for a JD.

 INPUTS
jd  - This is the given JD.

 RESULT
day - Day of the date.
month - Month of the date.
year  - Year of the date.

 EXAMPLE
...
date_JDToGregorian(2299161,&day,&month,&year);
...

 NOTES
It is better to use this function only from 1718867 to 2889835!

 BUGS
unknown.

 SEE ALSO

              JDToJulian()
              ,
              JDToHeis()
```

1.55 Date/JDToHeis

```
                  NAME
JDToHeis -- Returns the Heis date for a JD. (V33.095)
```

```
 SYNOPSIS
date_JDToHeis(jd,day,month,year);
        d0 a0   a1    a2

void date_JDToHeis(const unsigned long jd, unsigned short *const day,
    unsigned short *const month, long *const year);

void date_JDToHeis(const unsigned long jd, unsigned short &day,
    unsigned short &month, long &year);

 FUNCTION
Returns the Heis date for a JD.

 INPUTS
jd  - This is the given JD.

 RESULT
day - Day of the date.
month - Month of the date.
year  - Year of the date.

 EXAMPLE
...
date_JDToHeis(2299161,&day,&month,&year);
...

 NOTES
At the moment this is only a dummy to date_JDToGregorian, so:
It is better to use this function only from 1718867 to 2889835!

 BUGS
unknown.

 SEE ALSO

              JDToJulian()
              ,
              JDToGregorian()
```

1.56 Date/JDToJulian

```
                 NAME
JDToJulian -- Returns the Julian date for a JD. (V33.095)

 SYNOPSIS
date_JDToJulian(jd,day,month,year);
    d0 a0 a1    a2

void date_JDToJulian(const unsigned long jd, unsigned short *const day,
    unsigned short *const month, long *const year);

void date_JDToJulian(const unsigned long jd, unsigned short &day,
    unsigned short &month, long &year);
```

```
 FUNCTION
Returns the Julian date for a JD.

 INPUTS
jd  - This is the given JD.

 RESULT
day - Day of the date.
month - Month of the date.
year  - Year of the date.

 EXAMPLE
...
date_JDToJulian(2299160,&day,&month,&year);
...

 NOTES
It is better to use this function only from 1718867 to 2299160!

 BUGS
unknown.

 SEE ALSO

              JDToGregorian()
              ,
              JDToHeis()
```

1.57 Date/JDtoMJD

```
                  NAME
JDtoMJD -- Switches from JD to MJD. (V33)

 SYNOPSIS
mjd = date_JDtoMJD(jd);
d0       d0

unsigned long date_JDtoMJD(const unsigned long jd);

 FUNCTION
Returns the Modified Julianday of a Julianday.

 INPUTS
jd - Julianday

 RESULT
mjd - The Modified Julianday

 EXAMPLE
...
mjd = date_JDtoMJD(2449354);
...
```

```
 NOTES
none

 BUGS
Only use this function for jd > 2400001, because mjd is only
defined for this, otherwise system will crash!

 SEE ALSO

              MJDtoJD()
```

1.58 Date/JDToTime

```
                  NAME
JDToTime -- Returns the real time for a JD time. (V33)

 SYNOPSIS
time_JDToTime(jd,rhour,rmin,rsec);
        d0  a0  a1   a2

void time_JDToTime(float jd, unsigned short *const rhour,
    unsigned short *const rmin, unsigned short *const rsec);

void time_JDToTime(float jd, unsigned short &rhour,
    unsigned short &rmin, unsigned short &rsec);

 FUNCTION
Returns the real time for a JD time.

 INPUTS
jd - JD time

 RESULT
rhour - 24 hour real time
rmin  - real minutes
rsec  - real seconds

 EXAMPLE
...
time_JDToTime(0.76543,&rhour,&rmin,&rsec);
...

 NOTES
none.

 BUGS
If jd is > 0 (including days) there will be occur arithmetic bugs!

 SEE ALSO

              TimeToJD()
```

1.59 Date/JSYearToJD

```
                  NAME
JSYearToJD -- Calcs the JD from a Scaliger year. (V33)

 SYNOPSIS
jd = date_JSYearToJD(syear);
d0          d0

unsigned long date_JSYearToJD(const unsigned long syear);

 FUNCTION
Returns the Julianday of a Scaliger year.

 INPUTS
syear   - Scaliger year

 RESULT
jd - The Julianday

 EXAMPLE
...
jd = date_JSYearToJD(4800);
...

 NOTES
It is better to use this function only from 4707 to 6295!

 BUGS
unknown.

 SEE ALSO

              GSYearToJD()
              ,
              HSYearToJD()
```

1.60 Date/JulianDayDiff

```
                  NAME
JulianDayDiff -- Calculates the days between 2 dates. (V33)

 SYNOPSIS
days = date_JulianDayDiff(day1,month1,year1,day2,month2,year2);
 d0        d0  d1    d2    d3    d4  d5

long date_JulianDayDiff(const unsigned short day1, unsigned short month1,
    long year1, const unsigned short day2, unsigned short month2,
    long year2);

 FUNCTION
JulianDayDiff gives you back the number of days between
two specified dates.
```

```
 INPUTS
day1   - day of the first date
month1 - month of the first date
year1  - year of the first date
day2   - day of the second date
month2 - month of the second month
year2  - year of the second date

 RESULT
days - The number of days between the two dates
    (positive if date1 <= date2).

 EXAMPLE
long days;
...
days = date_JulianDayDiff(18,9,1970,22,1,1994);
printf("Age of Kai Hofmann in days : %ld\n",days);
...

 NOTES
It is better only to use this function for years from 8 to 1582!

 BUGS
No known bugs.

 SEE ALSO

              JulianLeapYear()
              ,
              JulianMonthDays()
              ,
              JulianYearDays()
              ,

              GregorianDayDiff()
              ,
              HeisDayDiff()
```

1.61 Date/JulianDaysAfterWeekday

```
                 NAME
JulianDaysAfterWeekday -- Returns the diff to the wday after. (V33)

 SYNOPSIS
days = date_JulianDaysAfterWeekday(day,month,year,weekday);
 d0           d0   d1   d2     d3

unsigned short date_JulianDaysAfterWeekday(const unsigned short day,
    const unsigned short month, const long year,
    const date_Weekdays weekday);

 FUNCTION
Returns the days to the weekday after the specified date.
```

```
If you specify the 22.1.1994 (Saturday) and Thursday
you get back 5!
If you specify the 22.1.1994 and Saturday you get back 0
(the same day)!

 INPUTS
day - day of the date
month - month of the date
year  - year of the date
weekday - weekday to search for building difference

 RESULT
days - The days after to the searched weekday.

 EXAMPLE
...
days = date_JulianDaysAfterWeekday(22,1,1994,Thursday);
...

 NOTES
It is better to use this function only from 8 to 1582!

 BUGS
See JulianWeekday()!

 SEE ALSO

              JulianWeekday()
              ,
              GregorianDaysAfterWeekday()
              ,
              HeisDaysAfterWeekday()
```

1.62 Date/JulianDaysBeforeWeekday

```
                  NAME
JulianDaysBeforeWeekday -- Returns the diff to the wday before. (V33)

 SYNOPSIS
days = date_JulianDaysBeforeWeekday(day,month,year,weekday);
 d0            d0  d1    d2    d3

unsigned short date_JulianDaysBeforeWeekday(const unsigned short day,
    const unsigned short month, const long year,
    const date_Weekdays weekday);

 FUNCTION
Returns the days to the weekday before the specified date.
If you specify the 22.1.1994 (Saturday) and Thursday
you get back 2!
If you specify the 22.1.1994 and Saturday you get back 0
(the same day)!

 INPUTS
```

```
day - day of the date
month - month of the date
year  - year of the date
weekday - weekday to search for building difference

 RESULT
days - The days gets you back to the searched weekday (0-6)
    If you get back an 8 an error occurs!

 EXAMPLE
...
days = date_JulianDaysBeforeWeekday(22,1,1994,Thursday);
...

 NOTES
It is better to use this function only from 8 to 02.1582!

 BUGS
See JulianWeekday()!

 SEE ALSO

              JulianWeekday()
              ,
              GregorianDaysBeforeWeekday()
              ,
              HeisDaysBeforeWeekday()
```

1.63 Date/JulianDiffDate

```
                  NAME
JulianDiffDate -- Returns the date for a diff to another date. (V33)

 SYNOPSIS
date_JulianDiffDate(day,month,year,days,dday,dmonth,dyear);
         d0   d1   d2   d3   a0    a1     a2

void date_JulianDiffDate(const unsigned short day,
    const unsigned short month, const long year, long days,
    unsigned short *const dday, unsigned short *const dmonth,
    long *const dyear);

void date_JulianDiffDate(const unsigned short day,
    const unsigned short month, const long year, long days,
    unsigned short &dday, unsigned short &dmonth, long &dyear);

 FUNCTION
Returns the date which lies days before/after the specified date.

 INPUTS
day   - day of the date
month - month of the date
year  - year of the date
days  - difference to the date in days
```

```
 RESULT
dday   - Destination day
dmonth - Destination month
dyear  - Destination year

 EXAMPLE
...
date_JulianDiffDate(23,1,1994,7,&dday,&dmonth,&dyear);
...

 NOTES
This function is OBSOLETE - please use
              JulianDiffDateRange()
                instead!
It is better to use this function only from 8 to 1582!

 BUGS
unknown.

 SEE ALSO

              JulianDayDiff()
              ,
              JulianMonthDays()
              ,
              GregorianDiffDate()
              ,
              HeisDiffDate()
```

1.64 Date/JulianDiffDateRange

```
                  NAME
JulianDiffDateRange -- Calc new date from old one and diff. (V33.296)

 SYNOPSIS
date_JulianDiffDateRange(day,month,year,days,months,
    years,dday,dmonth,dyear);
        d0   d1   d2   d3    d4
     d5    a0  a1    a2

void date_JulianDiffDateRange(const unsigned short day,
    const unsigned short month, const long year, long days,
    long months, long years, unsigned short *const dday,
    unsigned short *const dmonth, long *const dyear);

void date_JulianDiffDateRange(const unsigned short day,
    const unsigned short month, const long year, long days,
    long months, long years, unsigned short &dday,
    unsigned short &dmonth, long &dyear);

 FUNCTION
Returns the date which lies diffrange before/after the specified date.
```

```
 INPUTS
day    - day of the date
month  - month of the date
year   - year of the date
days   - difference to the date in days
months - difference to the date in months
years  - difference to the date in years

 RESULT
dday   - Destination day
dmonth - Destination month
dyear  - Destination year

 EXAMPLE
...
date_JulianDiffDateRange(23,1,1994,0,0,1,&dday,&dmonth,&dyear);
...

 NOTES
It is better to use this function only from 8 to 1582!

 BUGS
unknown.

 SEE ALSO

              JulianDiffDate()
              ,
              JulianRangeDiff()
              ,
              GregorianDiffDateRange()
              ,

              HeisDiffDateRange()
```

1.65 Date/JulianEaster

```
                  NAME
JulianEaster -- Returns the date of Easter in a year (V33.097)

 SYNOPSIS
date_JulianEaster(year,dday,dmonth);
       d0 a0    a1

void date_JulianEaster(const long year, unsigned short *const dday,
    unsigned short *const dmonth);

void date_JulianEaster(const long year, unsigned short &dday,
    unsigned short &dmonth);

 FUNCTION
Returns the date of Easter for a specified year.

 INPUTS
```

```
year  - Easter is calculated for this year

 RESULT
dday   - day of Easter Sunday
dmonth - month of Easter Sunday

 EXAMPLE
...
date_JulianEaster(1994,&dday,&dmonth);
...

 NOTES
Use this only for 31 to 1582!

 BUGS
None.

 SEE ALSO

              GregorianEaster()
              ,
              HeisEaster()
```

1.66 Date/JulianLastMonthDay

```
                  NAME
JulianLastMonthDay -- The number of last day in a month. (V33.234)

 SYNOPSIS
day = date_JulianLastMonthDay(month,year);
 d0            d0    d1

unsigned short date_JulianLastMonthDay(const unsigned short month,
    const long year);

 FUNCTION
JulianLastMonthDay returns the number of the last day for a given
month.

 INPUTS
month - The month from which you want to get the last day.
year  - The year in which the month is.

 RESULT
day - The number of the last day the month uses, or 0 if you use
    an incorrect month.

 EXAMPLE
...
day = date_JulianLastMonthDay(1,1994); /* 31 */
...

 NOTES
It is better only to use this function for years from 8 to 09.1582!
```

```
 BUGS
No known bugs.

 SEE ALSO

              JulianMonthDays()
              ,
              GregorianLastMonthDay()
              ,
              HeisLastMonthDay()
```

1.67 Date/JulianLeapYear

```
                  NAME
JulianLeapYear -- Checks if a year is a leap year. (V33)

 SYNOPSIS
leapyear = date_JulianLeapYear(year);
   d0       d0

bool date_JulianLeapYear(const long year);

 FUNCTION
JulianLeapYear checks if a year is a leap year in the Julian calendar
For years after Chr. it checks if the year is devideable by 4.
For years before Chr. a leap year must have a modulo 4 value of 1

 INPUTS
year - The year which should be checked (from -32768 to 32767)
    I think only values from 8 to 32767 are valid, because of
    the variant that was done by Augustus!

 RESULT
leapyear - true if the year is a leap year, otherwise false.

 EXAMPLE
...
if (date_JulianLeapYear(1994))
 {
  printf("leap year!\n");
 }
else
 {
  printf("no leap year!\n");
 }
...

 NOTES
A year is 365.25 days long!
Use this function only for values from 8 to 1582!

 BUGS
No known bugs.
```

```
 SEE ALSO

              GregorianLeapYear()
              ,
              HeisLeapYear()
```

1.68 Date/JulianMonthDays

```
                  NAME
JulianMonthDays -- Returns the number of days of a month. (V33)

 SYNOPSIS
days = date_JulianMonthDays(month,year);
 d0          d0    d1

unsigned short date_JulianMonthDays(const unsigned short month,
    const long year);

 FUNCTION
JulianMonthDays returns the number of days a month in
a specified year has.

 INPUTS
month - The month from which you want to get the number of days.
year  - The year in which the month is.

 RESULT
days - The number of days the month uses, or 0 if you use
    an incorrect month.

 EXAMPLE
unsigned short days;
...
days = date_JulianMonthDays(1,1994);
printf("Days of January 1994 : %hu\n",days);
...

 NOTES
It is better only to use this function for years from 8 to 09.1582!

 BUGS
No known bugs.

 SEE ALSO

              JulianLeapYear()
              ,
              GregorianMonthDays()
              ,
              HeisMonthDays()
```

1.69 Date/JulianRangeDiff

```
                  NAME
JulianRangeDiff -- Calculates the range between 2 dates. (V33.297)

 SYNOPSIS
date_JulianRangeDiff(day1,month1,year1,day2,month2,year2,days,months,
    years);
          d0    d1    d2  d3    d4    d5    a0  a1
     a2

void date_JulianRangeDiff(unsigned short day1,
    unsigned short month1, long year1, unsigned short day2,
    unsigned short month2, long year2, short *const days,
    short *const months, long *const years);

void date_JulianRangeDiff(unsigned short day1,
    unsigned short month1, long year1, unsigned short day2,
    unsigned short month2, long year2, short &days, short &months,
    long &years);

 FUNCTION
JulianRangeDiff gives you back the number of days, months and years
between two specified dates.

 INPUTS
day1   - day of the first date
month1 - month of the first date
year1  - year of the first date
day2   - day of the second date
month2 - month of the second month
year2  - year of the second date

 RESULT
days   - The number of days between the two dates
   (positive if date1 <= date2).
months - The number of months between the two dates
   (positive if date1 <= date2).
years  - The number of years between the two dates
   (positive if date1 <= date2).

 EXAMPLE
short days,months;
long years;
...
date_JulianRangeDiff(18,9,1970,25,1,1998,&days,&months,&years);
printf("Age of Kai Hofmann is : %ld years, %hd months, %hd days\n",
   years,months,days);
...

 NOTES
It is better only to use this function for years from 8 to 1582!

 BUGS
No known bugs.
```

```
 SEE ALSO

              JulianDayDiff()
              ,
              JulianDiffDateRange()
              ,
              GregorianRangeDiff()
              ,

              HeisRangeDiff()
```

1.70 Date/JulianToJD

```
                  NAME
JulianToJD -- Returns the JD for a date. (V33)

 SYNOPSIS
jd = date_JulianToJD(day,month,year);
d0          d0   d1  d2

unsigned long date_JulianToJD(const unsigned short day,
    const unsigned short month, const long year);

 FUNCTION
Returns the JD for a Julian date.

 INPUTS
day  - day of the date to convert
month  - month of the date to convert
year   - year of the date to convert

 RESULT
jd - This is the JD

 EXAMPLE
...
jd = date_JulianToJD(23,1,1994);
...

 NOTES
It is better to use this function only from 8 to 1582!

 BUGS
unknown.

 SEE ALSO

              JSYearToJD()
              ,
              JYearToScaliger()
              ,
              JulianDayDiff()
              ,
              GregorianToJD()
```

,

HeisToJD()

1.71 Date/JulianWeek

```
                  NAME
JulianWeek -- Gets the number of the week for a specified date. (V33)

 SYNOPSIS
weeknr = date_JulianWeek(day,month,year);
  d0        d0  d1   d2

unsigned short date_JulianWeek(const unsigned short day,
    const unsigned short month, const long year);

 FUNCTION
JulianWeek gets the number of the week for a specified date.

 INPUTS
day   - day of the date
month - month of the date
year  - year of the date

 RESULT
week - This is the number of the week the specified date lies in.
    If the first day in a new year is a Friday, Saturday or
    Sunday, this would be the last week of the last year!
    If the 29.12. is a Monday, the 30.12. is a Monday or a Tuesday,
    the 31.12. is a Monday, Tuesday or a Wednesday this is the
    first week of the next year!

 EXAMPLE
...
weeknr = date_JulianWeek(4,10,1582);
...

 NOTES
It is is better only to use this function for years from 8 to 1582!

 BUGS
For years < 0 errors could occur.

 SEE ALSO

              GregorianWeek()
              ,
              HeisWeek()
              ,
              JulianWeekday()
              ,
              JulianDayDiff()
              ,
```

```
               SetFirstWeekday()
```

1.72 Date/JulianWeekday

```
                  NAME
JulianWeekday -- Gets the weekday of a specified date. (V33)

 SYNOPSIS
weekday = date_JulianWeekday(day,month,year);
  d0            d0   d1 d2

date_Weekdays date_JulianWeekday(const unsigned short day,
    unsigned short month, long year);

 FUNCTION
JulianWeekday gets the weekday for a specified date.

 INPUTS
day   - day of the date
month - month of the date
year  - year of the date

 RESULT
weekday - This result is of type:
    date_Weekdays = {date_dayerr,date_Monday,date_Tuesday,
         date_Wednesday,date_Thursday,date_Friday,
         date_Saturday,date_Sunday};
    dayerr will show you, that an error occurs!

 EXAMPLE
...
weekday = date_JulianWeekday(4,10,1582);
if (weekday == dayerr)
 {
  ...
 }
...

 NOTES
It is better only to use this function for years from 8 to 4.10.1582!
In this version no dayerr will occur!

 BUGS
For years <= 0 errors could occur, or systemcrashs(?).

 SEE ALSO

              GregorianWeekday()
              ,
              HeisWeekday()
```

1.73 Date/JulianWWtoDM

```
                  NAME
JulianWWtoDM -- Convert Weekday,Week to Day,Month. (V33.243)

 SYNOPSIS
date_JulianWWtoDM(weekday,week,year,dday,dmonth,dyear);
         d0    d1 d2   a0    a1   a2

void date_JulianWWtoDM(const date_Weekdays weekday,
   const unsigned short week, const long year,
   unsigned short *const dday, unsigned short *const dmonth,
   long *const dyear);

void date_JulianWWtoDM(const date_Weekdays weekday,
   const unsigned short week, const long year,
   unsigned short &dday, unsigned short &dmonth,
   long &dyear);

 FUNCTION
Converts weekday,week to day,month.

 INPUTS
weekday  - weekday of the date to convert
week   - week of the date to convert
year   - year of the date to convert

 RESULT
day   - day of the converted date
month - month of the converted date
year  - year of the converted date

 EXAMPLE
unsigned short day,month;
long year;
...
date_JulianWWtoDM(date_Thursday,14,1997,&day,&month,&year);
/* 1997-04-03 */
...

 NOTES
It is better to use this function only from 8 to 1582!
Be careful, the resulting year might be different from the input
year!

 BUGS
unknown.

 SEE ALSO

              GregorianWWtoDM()
              ,
              HeisWWtoDM()
```

1.74 Date/JulianYearDays

```
                  NAME
JulianYearDays -- Gives back the number of days in a year. (V33)

 SYNOPSIS
days = date_JulianYearDays(year);
 d0         d0

unsigned short date_JulianYearDays(const long year);

 FUNCTION
JulianYearDays gives you back the number of days in
a specified year.

 INPUTS
year  - The year in which to count the days.

 RESULT
days - The number of days the year uses.

 EXAMPLE
unsigned short days;
...
days = date_JulianYearDays(1994);
printf("Days of 1994 : %hu\n",days);
...

 NOTES
It is better only to use this function for years from 8 to 1581!

 BUGS
No known bugs.

 SEE ALSO

              JulianMonthDays()
              ,
              GregorianYearDays()
              ,
              HeisYearDays()
```

1.75 Date/JYearToScaliger

```
                  NAME
JYearToScaliger -- Returns the year as Scaliger year. (V33)

 SYNOPSIS
syear = date_JYearToScaliger(year);
 d0           d0

unsigned long date_JYearToScaliger(const long year);
```

```
 FUNCTION
Returns the Scaliger year.

 INPUTS
year   - Julian year

 RESULT
syear - The Scaliger year

 EXAMPLE
...
syear = date_JYearToScaliger(1582);
...

 NOTES
It is better to use this function only from 8 to 1582!

 BUGS
unknown.

 SEE ALSO

              GYearToScaliger()
              ,
              HYearToScaliger()
```

1.76 Date/LastMonthDay

```
                  NAME
LastMonthDay -- The number of last day in a month. (V33.310)

 SYNOPSIS
day = date_LastMonthDay(month,year,calendar);
 d0       d0  d1   d2

unsigned short date_LastMonthDay(const unsigned short month,
    const long year, const date_Calendars calendar);

 FUNCTION
LastMonthDay returns the number of the last day for a given
month.

 INPUTS
month    - The month from which you want to get the last day.
year     - The year in which the month is.
calendar - Calendar system to use

 RESULT
days - The number of the last day the month uses, or 0 if you use
    an incorrect month.

 EXAMPLE
...
day = date_LastMonthDay(1,1994,date_Gregorian); /* 31 */
```

```
...

 NOTES
Use this function only for years from 8 to 3199!

 BUGS
none.

 SEE ALSO

              JulianLastMonthDay()
              ,
              GregorianLastMonthDay()
              ,
              HeisLastMonthDay()
```

1.77 Date/LeapYear

```
                  NAME
LeapYear -- Checks if a year is a leap year. (V33.310)

 SYNOPSIS
leapyear = date_LeapYear(year,calendar);
   d0        d0   d1

bool date_LeapYear(const long year, const date_Calendars calendar);

 FUNCTION
LeapYear checks if a year is a leap year.

 INPUTS
year     - The year which should be checked (from -32768 to 32767)
           I think only values from 8 to 3200 are valid, because of
           the variant that was done by Augustus!
calendar - Calendar system to use

 RESULT
leapyear - true if the year is a leap year, otherwise false.

 EXAMPLE
...
if (date_LeapYear(1994,date_Gregorian))
 {
  printf("leap year!\n");
 }
else
 {
  printf("no leap year!\n");
 }
...

 NOTES
Use this function only for values from 8 to 8000!
```

```
 BUGS
No known bugs.

 SEE ALSO

              JulianLeapYear()
              ,
              GregorianLeapYear()
              ,
              HeisLeapYear()
```

1.78 Date/LMT

```
 NAME
LMT -- Calculates your local time in your timezone (V33)

 SYNOPSIS
secs = time_LMT(secs,meridian,pos);
 d0    d0 d1    d2

unsigned long time_LMT(const unsigned long secs,
    const float meridiandegree, const float posdegree);

 FUNCTION
Calculates your Local Mean Time of your place!

 INPUTS
secs   - Seconds of the running day (hours*3600+min*60+sec)
meridian - Degrees of your timezone-meridian
pos  - Degrees of your place

 RESULT
secs - Local seconds of the running day

 EXAMPLE
...
secs = time_LMT(76080,-15.0,-8.923055556);
...

 NOTES
none

 BUGS
No errorcheck, if you put in valid degrees (-180 to +180)

 SEE ALSO
```

1.79 Date/LocalToGMT

```
                NAME
LocalToGMT -- Converts a local time to GMT (V33.300)

 SYNOPSIS
datetime_LocalToGMT(ljd,lsecs,DST,zonemin,gjd,gsecs);
         d0   d1  d2    d3    a0   a1

void datetime_LocalToGMT(const unsigned long ljd,
    const unsigned long lsecs, const bool DST,
    const short zonemin, unsigned long *const gjd,
    unsigned long *const gsecs);

void datetime_LocalToGMT(const unsigned long ljd,
    const unsigned long lsecs, const bool DST,
    const short zonemin, unsigned long &gjd,
    unsigned long &gsecs);

 FUNCTION
Converts a local date/time pair into a GMT date/time pair.
The conversion considers the local daylight savings time.

 INPUTS
ljd    - Local Julian Date
lsecs    - Local time in seconds
DST    - Daylight saving status
zonemin    - Time that was added to GMT time to get the local time
       zone (in minutes -779 to +779)

 RESULT
gjd   - GMT Julian Date
gsecs - GMT time in seconds

 EXAMPLE
unsigned long gjd,gsecs;
...
datetime_LocalToGMT(2450919,37500,true,+60,&gjd,&gsecs);
...

 NOTES
None.

 BUGS
No errorcheck, if you use a valid date/time.

 SEE ALSO

             GMTToLocal()
             ,
             JulianToJD()
             ,
             GregorianToJD()
             ,
             HeisToJD()
             ,

             JDToJulian()
```

```
,
JDToGregorian()
,
JDToHeis()
,
TimeToSec()
,
SecToTime()
,

TimeZoneFactor()
```

1.80 Date/MJDtoJD

```
                  NAME
MJDtoJD -- Switches from MJD to JD. (V33)

 SYNOPSIS
jd = date_MJDtoJD(mjd);
d0       d0

unsigned long date_MJDtoJD(const unsigned long mjd);

 FUNCTION
Returns the Julianday of a Modified Julianday.

 INPUTS
mjd - Modified Julianday

 RESULT
jd - The Julianday

 EXAMPLE
...
jd = date_MJDtoJD(49353);
...

 NOTES
none

 BUGS
unknown.

 SEE ALSO

              JDtoMJD()
```

1.81 Date/MonthDays

```
                NAME
MonthDays -- Returns the number of days of a month. (V33.310)

 SYNOPSIS
days = date_MonthDays(month,year,calendar);
 d0     d0    d1    d2

unsigned short date_MonthDays(const unsigned short month,
    const long year, const date_Calendars calendar);

 FUNCTION
MonthDays returns the number of days a month in
a specified year has.
For the year 1582 and the month 10 there are only 21 days,
because of the -reform 10 days are deleted from
the month (for more - look out for books about this!)

 INPUTS
month    - The month from which you want to get the number of days.
year     - The year in which the month is.
calendar - Calendar system to use

 RESULT
days - The number of days the month uses, or 0 if you use
    an incorrect month.

 EXAMPLE
unsigned short days;
...
days = date_MonthDays(1,1994,date_Gregorian);
printf("Days of January 1994 : %hu\n",days);
...

 NOTES
Use this function only for years from 8 to 8000!

 BUGS
none.

 SEE ALSO

             JulianMonthDays()
             ,
             GregorianMonthDays()
             ,
             HeisMonthDays()
```

1.82 Date/MonthShortText

```
                NAME
MonthShortText -- Get the month as short text string. (V33.092)

 SYNOPSIS
```

```
maxlen = date_MonthShortText(month,mtext,lang);
 d0           d0   a0  d1

unsigned short date_MonthShortText(const unsigned short month,
    char *const mtext, const date_Languages lang);

 FUNCTION
This function gets the short text string for the month-number.

 INPUTS
month - Month to transform into a string.
mtext - Pointer to a string to fill in the short month-text.
lang  - Language for which you want the short month-text.

 RESULT
maxlen - Maximum possible length for the short month-string, this
    should help you if you want to justify the string right or if
    you want to center it (Normal is three!).
    0 indicates an error!

 EXAMPLE
...
char mtxt[4];
...
maxlen = date_MonthShortText(12,&mtxt,English);
...

 NOTES
Languages:
Locale    : This is an Amiga >= OS2.1 only feature, for <= OS2.0
      and other systems it will return English text!

 BUGS
In this version there is no check, if there is enough space in
wtext!

 SEE ALSO

              WeekdayText()
              ,
              WeekdayShortText()
              ,
              MonthText()
```

1.83 Date/MonthText

```
                 NAME
MonthText -- Get the month as text string. (V33.091)

 SYNOPSIS
maxlen = date_MonthText(month,mtext,lang);
 d0      d0   a0    d1

unsigned short date_MonthText(const unsigned short month,
```

```
    char *const mtext, const date_Languages lang);

 FUNCTION
This function gets the text string for the month-number.

 INPUTS
month - Month to transform into a string.
mtext - Pointer to a string to fill in the month-text.
lang  - Language for which you want the month-text.

 RESULT
maxlen - Maximum possible length for the month-string, this should
    help you if you want to justify the string right or if you
    want to center it!
    0 indicates an error!

 EXAMPLE
...
char mtxt[20];
...
maxlen = date_MonthText(12,&mtxt,English);
...

 NOTES
Languages:
Locale    : This is an Amiga >= OS2.1 only feature, for <= OS2.0
      and other systems it will return English text!

 BUGS
In this version there is no check, if there is enough space in
wtext!

 SEE ALSO

              WeekdayText()
              ,
              WeekdayShortText()
              ,
              MonthShortText()
```

1.84 Date/NextValidDate

```
                 NAME
NextValidDate -- Returns the next valid date (V33.310)

 SYNOPSIS
date_NextValidDate(day,month,year,dday,dmonth,dyear,calendar);
        d0   d1  d2    a0   a1   a2   d3

void date_NextValidDate(const unsigned short day,
    const unsigned short month, const long year,
    unsigned short *const dday, unsigned short *const dmonth,
    long *const dyear, const date_Calendars calendar);
```

```
void date_NextValidDate(const unsigned short day,
    const unsigned short month, const long year,
    unsigned short &dday, unsigned short &dmonth, long &dyear,
    const date_Calendars calendar);

 FUNCTION
Returns the next valid date after a given one.

 INPUTS
day      - Day of the date.
month    - Month of the date.
year     - Year of the date.
calendar - Calendar system to use

 RESULT
dday   - Day of the next valid date
dmonth - Month of the next valid date
dyear  - Year of the next valid date

 EXAMPLE
...
date_NextValidDate(29,2,2000,&day,&month,&year,date_Gregorian);
...

 NOTES
None.

 BUGS
None.

 SEE ALSO

              ValidDate()
              ,
              NextValidJulianDate()
              ,
              NextValidGregorianDate()
              ,

              NextValidHeisDate()
              ,
              PreviousValidDate()
```

1.85 Date/NextValidGregorianDate

```
                  NAME
NextValidGregorianDate -- Returns the next valid date (V33.236)

 SYNOPSIS
date_NextValidGregorianDate(day,month,year,dday,dmonth,dyear);
           d0   d1  d2    a0    a1   a2

void date_NextValidGregorianDate(const unsigned short day,
   const unsigned short month, const long year,
```

```
   unsigned short *const dday, unsigned short *const dmonth,
   long *const dyear);

void date_NextValidGregorianDate(const unsigned short day,
   const unsigned short month, const long year,
   unsigned short &dday, unsigned short &dmonth, long &dyear);

 FUNCTION
Returns the next valid date after a given one.

 INPUTS
day   - Day of the date.
month - Month of the date.
year  - Year of the date.

 RESULT
dday   - Day of the next valid date
dmonth - Month of the next valid date
dyear  - Year of the next valid date

 EXAMPLE
...
date_NextValidGregorianDate(29,2,2000,&day,&month,&year);
...

 NOTES
None.

 BUGS
None.

 SEE ALSO

              ValidGregorianDate()
              ,
              NextValidJulianDate()
              ,
              NextValidHeisDate()
              ,

              PreviousValidGregorianDate()
```

1.86 Date/NextValidHeisDate

```
                  NAME
NextValidHeisDate -- Returns the next valid date (V33.235)

 SYNOPSIS
date_NextValidHeisDate(day,month,year,dday,dmonth,dyear);
      d0   d1  d2    a0    a1     a2

void date_NextValidHeisDate(const unsigned short day,
   const unsigned short month, const long year,
   unsigned short *const dday, unsigned short *const dmonth,
```

```
   long *const dyear);

void date_NextValidHeisDate(const unsigned short day,
   const unsigned short month, const long year,
   unsigned short &dday, unsigned short &dmonth, long &dyear);

 FUNCTION
Returns the next valid date after a given one.

 INPUTS
day   - Day of the date.
month - Month of the date.
year  - Year of the date.

 RESULT
dday   - Day of the next valid date
dmonth - Month of the next valid date
dyear  - Year of the next valid date

 EXAMPLE
...
date_NextValidHeisDate(29,2,2000,&day,&month,&year);
...

 NOTES
None.

 BUGS
None.

 SEE ALSO

              ValidHeisDate()
              ,
              NextValidJulianDate()
              ,
              NextValidGregorianDate()
              ,

              PreviousValidHeisDate()
```

1.87 Date/NextValidJulianDate

```
                  NAME
NextValidJulianDate -- Returns the next valid date (V33.236)

 SYNOPSIS
date_NextValidJulianDate(day,month,year,dday,dmonth,dyear);
        d0   d1  d2  a0    a1     a2

void date_NextValidJulianDate(const unsigned short day,
   const unsigned short month, const long year,
   unsigned short *const dday, unsigned short *const dmonth,
   long *const dyear);
```

```
void date_NextValidJulianDate(const unsigned short day,
   const unsigned short month, const long year,
   unsigned short &dday, unsigned short &dmonth, long &dyear);

 FUNCTION
Returns the next valid date after a given one.

 INPUTS
day   - Day of the date.
month - Month of the date.
year  - Year of the date.

 RESULT
dday   - Day of the next valid date
dmonth - Month of the next valid date
dyear  - Year of the next valid date

 EXAMPLE
...
date_NextValidJulianDate(29,2,2000,&day,&month,&year);
...

 NOTES
None.

 BUGS
None.

 SEE ALSO

              ValidJulianDate()
              ,
              NextValidGregorianDate()
              ,
              NextValidHeisDate()
              ,

              PreviousValidJulianDate()
```

1.88 Date/NumToDate

```
                  NAME
NumToDate -- Returns the real date for a numeric time. (V33.250)

 SYNOPSIS
date_NumToDate(num,day,month,year);
         d0  a0 a1    a2

void date_NumToDate(long num, unsigned short *const day,
    unsigned short *const month, long *const year);

void date_NumToDate(long num, unsigned short &day,
    unsigned short &month, long &year);
```

```
 FUNCTION
Returns the real date for a numeric time.

 INPUTS
num - Date in numeric format

 RESULT
day - Day of the date.
month - Month of the date.
year  - Year of the date.

 EXAMPLE
...
date_NumToDate(19970628,&day,&month,&year);
...

 NOTES
Negative years will be handled correctly.

 BUGS
None.

 SEE ALSO

              DateToNum()
```

1.89 Date/NumToTime

```
                  NAME
NumToTime -- Returns the real time for a numeric time. (V33.250)

 SYNOPSIS
time_NumToTime(num,rhour,rmin,rsec);
    d0  a0    a1   a2

void time_NumToTime(unsigned short num,
    unsigned short *const rhour, unsigned short *const rmin,
    unsigned short *const rsec);

void time_NumToTime(unsigne dshort num, unsigned short &rhour,
    unsigned short &rmin, unsigned short &rsec);

 FUNCTION
Returns the real time for a numeric time.

 INPUTS
num - Numeric time

 RESULT
rhour - hours
rmin  - minutes
rsec  - seconds
```

```
 EXAMPLE
...
time_NumToTime(123600,&rhour,&rmin,&rsec);
...

 NOTES
None.

 BUGS
None.

 SEE ALSO

              TimeToNum()
```

1.90 Date/ParseDate

```
                 NAME
ParseDate -- Parses a date string (V33.290)

 SYNOPSIS
pos = date_ParseDate(fmt,dstr,lang,calendar,actualjd,
   day,month,year,pfmt,plang);
          d0  d1  d2    d3       d4
   a0  a1    a2  a3  d5

short date_ParseDate(const char *const fmt, const char *const dstr,
   const date_Languages lang, const date_Calendars calendar,
   const unsigned long actualjd, unsigned short *const day,
   unsigned short *const month, long *const year, char *const pfmt,
   date_Languages *const plang);

short date_ParseDate(const char *const fmt, const char *const dstr,
   const date_Languages lang, const date_Calendars calendar,
   const unsigned long actualjd, unsigned short *const day,
   unsigned short *const month, long *const year, char *const pfmt,
   date_Languages &plang);

 FUNCTION
Parses the date from a string.

 INPUTS
fmt     - Format template string
dstr      - Datestring to parse
lang      - Language to use for strings
calendar    - Calendar system to use for decoding
actualjd    - Actual date in JD form
pfmt      - Buffer for format template of parsed string or NULL

 RESULT
pos -  0 : The string has been parsed successfully.
    >0 : An error occured during parsing - subtract one to get
         the position of the error within the string.
    -1 : Parameter error (one or more pointers are pointing to
```

```
         NULL)
    -2 : Format template string to long
day - Day of the date to format
month - Month of the date to format
year  - Year of the date to format
pfmt  - Format template of parsed string
plang - Parsed language

 EXAMPLE
short retval;
unsigned short day,month;
long year;
char pfmtbuf[20];
date_Languages lang;
...
retval = date_ParseDate("%Y-%m-%d","1997-04-03",date_Locale,
   date_Heis,2450809,&day,&month,&year,pfmtbuf,&lang);
...

 SYNTAX
Syntax of Amiga compatible % commands:
%d : Day number with leading 0s
%e : Day number with leading spaces
%m : Month number with leading 0s
%h : Abbreviated month name
%b : Abbreviated month name
%B : Month name
%y : Year using two digits with leading 0s
%Y : Year using four digits with leading 0s
%j : Julian date
%w : Weekday number
%a : Abbreviated weekday name
%A : Weekday name
%U : Week number, taking Sunday as first day of week
%W : Week number, taking Monday as first day of week
%x : Same as "%m/%d/%y"
%D : Same as "%m/%d/%y"

Syntax of % commands:
%Ddf  : Day with leading 0s
%Ddv  : Day without leading 0s
%DDf  : Day within the year with leading 0s
%DDv  : Day within the year without leading 0s
%Dmf  : Month with leading 0s
%Dmv  : Month without leading 0s
%Dms  : Month string
%Dma  : Abbreviated month string
%Dy2f : 2-digit year with leading 0s
%Dy2v : 2-digit year without leading 0s
%Dy4f : 4-digit year with leading 0s
%Dy4v : 4-digit year without leading 0s
%Dys  : Scaliger year
%Dj   : JD date
%DJ   : MJD date
%Dwn  : Weekday number (1-7)
%Dws  : Weekday string
%Dwa  : Abbreviated weekday string
```

```
%DWf  : Weeknumber with leading 0s
%DWv  : Weeknumber without leading 0s
%DMf  : Age of the moon (0-30 ?) with leading 0s
%DMv  : Age of the moon (0-30 ?) without leading 0s

 NOTES
%DMf and %DMv are only pseudo implementations, a date can not be
constructed with them.

The date-string autodetection knows over 40 date-string formats:

(yesterday|today|tomorrow)
wdn
wdn[,] mmm d[ (yy|yyyy)]
wdn[,] d[.][ ]mmm[,][ ][(yy|yyyy)]
wdn[,] d[.][ ]m[.][ ][(yy|yyyy)]
mmm d[ yyyy]
mmm/d[/(yy|yyyy)]
mmm-d[-(yy|yyyy)]
yyyy[-]Www[-]n
yyyy(-|/)m(-|/)d
yyyy(-|/)n[n[n]]
yy[-]Www[-]n
yy-nnn
d. mmm [(yy|yyyy)]
d.m.[(yy|yyyy)]
d-mmm[-(yy|yyyy)]
d-m[-yyyy]
         Detection condition:
yyyymmdd   | ddmmyyyy    (valid date test)
jjjjjjj    | yyyynnn     (1723980 >= JD <= 2914672)
JJJJJ
JJJJJJ     | yymmdd    (MJD <= 514671)
d-m-yy     | yy-m-d    (yy > 31 | valid date test)
d/m[/yyyy] | m/d/yyyy    (valid date test)
d/m/yy     | m/d/yy   | yy/m/d   (valid date test)

 BUGS
No known bugs.

 SEE ALSO

             FormatDate()
```

1.91 Date/ParseTime

```
                 NAME
ParseTime -- Parses a time string (V33.282)

 SYNOPSIS
pos = time_ParseTime(fmt,tstr,ChangeDay,ChangeHour,DST,hour,min,
   sec,zonemin,pfmt);
  d0         d0   d1   d2     d3     d4 a0   a1
   a2 a3  d5
```

```
short time_ParseTime(const char *const fmt,
   const char *const tstr, time_ChangeDay ChangeDay,
   unsigned short ChangeHour, bool *const DST,
   unsigned short *const hour, unsigned short *const min,
   unsigned short *const sec, short *const zonemin,
   char *const pfmt);

 FUNCTION
Parses the time from a string.

 INPUTS
fmt   - Format template string
      (max. length = 46 characters)
tstr    - Timestring to parse
ChangeDay - Normal day, winter to summer or summer to winter
      time change day.
ChangeHour  - Hour of summer/winter time change
DST   - Daylight saving status
zonemin   - Time that was added to GMT time to get the local
      time zone (in minutes -779 to +779)
pfmt    - Buffer for format template of parsed string or NULL

 RESULT
pos -  0 : The string has been parsed successfully.
    >0 : An error occured during parsing - subtract one to get
         the position of the error within the string.
    -1 : Parameter error (one or more pointers are pointing to
         NULL)
    -2 : Format template string to long
hour  - Hour of the time that was parsed
min - Minute of the time that was parsed
sec - Second of the time that was parsed
DST - Daylight saving status
zonemin - Time that was added to GMT time to get the local
    time zone (in minutes -779 to +779)
pfmt  - Format template of parsed string

 EXAMPLE
bool DST = false;
unsigned short Hour,Min,Sec;
short ZoneMin = 60,retval;
char pfmtbuf[20];
...
retval = time_ParseTime(NULL,"20:14",time_Normal,2,&DST,
   &Hour,&Min,&Sec,&ZoneMin,pfmtbuf);
...

 SYNTAX
Syntax of Amiga compatible format template % strings:
%q : Hour using 24-hour style
%H : Hour using 24-hour style with leading 0s
%Q : Hour using 12-hour style
%I : Hour using 12-hour style with leading 0s
%p : AM or PM strings
%M : The number of minutes with leading 0s
%S : Number of seconds with leadings 0s
```

```
%R : Same as "%H:%M"
%X : Same as "%H:%M:%S"
%T : Same as "%H:%M:%S"
%r : Same as "%I:%M:%S %p"

Syntax of format template % commands:
%Th1f  : 12 with leading 0s
%Th1v  : 12 without leading 0s
%Th2f  : 24 with leading 0s
%Th2v  : 24 without leading 0s
%Tpso  : a/p
%Tpsu  : A/P
%Tplo  : am/pm
%Tplu  : AM/PM
%Tmf   : with leading 0s
%Tmv   : without leading 0s
%Tsf   : with leading 0s
%Tsv   : without leading 0s
%Tj.   : starting with '.'
%Tj,   : starting with ','
%Tj0   : starting with '0.'
%Tj1   : starting with '0,'

%Tzh?? : hours only
%Tzm?  : 0100
%TzM?? : 01:00
%Tz?z? : Use Z   for UTC
%Tz?0? : Use +00 for UTC
%Tz??f : use leading 0s
%Tz??v : do not use leading 0s

%Tc1   : Use DST for s->w switch
%Tc2   : Use I/II for s->w switch
%Tc3   : Use a/b for s->w switch (24h only) - on 12h falls back to 2

 NOTES
None.

 BUGS
No known bugs.

 SEE ALSO

              FormatTime()
```

1.92 Date/PreviousValidDate

```
                  NAME
PreviousValidDate -- Returns the prev. valid date (V33.310)

 SYNOPSIS
date_PreviousValidDate(day,month,year,dday,dmonth,dyear,calendar);
       d0   d1  d2  a0    a1     a2    d3
```

```
void date_PreviousValidDate(const unsigned short day,
    const unsigned short month, const long year,
    unsigned short *const dday, unsigned short *const dmonth,
    long *const dyear, const date_Calendars calendar);

void date_PreviousValidDate(const unsigned short day,
    const unsigned short month, const long year,
    unsigned short &dday, unsigned short &dmonth, long &dyear,
    const date_Calendars calendar);

 FUNCTION
Returns the previous valid date before a given one.

 INPUTS
day      - Day of the date.
month    - Month of the date.
year     - Year of the date.
calendar - Calendar system to use

 RESULT
dday   - Day of the previous valid date
dmonth - Month of the previous valid date
dyear  - Year of the previous valid date

 EXAMPLE
...
date_PreviousValidDate(29,2,2000,&day,&month,&year,date_Gregorian);
...

 NOTES
None.

 BUGS
None.

 SEE ALSO

              ValidDate()
              ,
              PreviousValidJulianDate()
              ,
              PreviousValidGregorianDate()
              ,

              PreviousValidHeisDate()
              ,
              NextValidDate()
```

1.93 Date/PreviousValidGregorianDate

```
                  NAME
PreviousValidGregorianDate -- Returns the prev. valid date (V33.236)

 SYNOPSIS
```

```
date_PreviousValidGregorianDate(day,month,year,dday,dmonth,dyear);
         d0   d1  d2  a0    a1     a2

void date_PreviousValidGregorianDate(const unsigned short day,
   const unsigned short month, const long year,
   unsigned short *const dday, unsigned short *const dmonth,
   long *const dyear);

void date_PreviousValidGregorianDate(const unsigned short day,
   const unsigned short month, const long year,
   unsigned short &dday, unsigned short &dmonth, long &dyear);

 FUNCTION
Returns the previous valid date before a given one.

 INPUTS
day   - Day of the date.
month - Month of the date.
year  - Year of the date.

 RESULT
dday   - Day of the previous valid date
dmonth - Month of the previous valid date
dyear  - Year of the previous valid date

 EXAMPLE
...
date_PreviousValidGregorianDate(29,2,2000,&day,&month,&year);
...

 NOTES
None.

 BUGS
None.

 SEE ALSO

              ValidGregorianDate()
              ,
              PreviousValidJulianDate()
              ,

              PreviousValidHeisDate()
              ,
              NextValidGregorianDate()
```

1.94 Date/PreviousValidHeisDate

```
                  NAME
PreviousValidHeisDate -- Returns the previous valid date (V33.235)

 SYNOPSIS
date_PreviousValidHeisDate(day,month,year,dday,dmonth,dyear);
```

```
          d0   d1  d2    a0  a1 a2

void date_PreviousValidHeisDate(const unsigned short day,
   const unsigned short month, const long year,
   unsigned short *const dday, unsigned short *const dmonth,
   long *const dyear);

void date_PreviousValidHeisDate(const unsigned short day,
   const unsigned short month, const long year,
   unsigned short &dday, unsigned short &dmonth, long &dyear);

 FUNCTION
Returns the previous valid date before a given one.

 INPUTS
day   - Day of the date.
month - Month of the date.
year  - Year of the date.

 RESULT
dday   - Day of the previous valid date
dmonth - Month of the previous valid date
dyear  - Year of the previous valid date

 EXAMPLE
...
date_PreviousValidHeisDate(29,2,2000,&day,&month,&year);
...

 NOTES
None.

 BUGS
None.

 SEE ALSO

             ValidHeisDate()
             ,
             PreviousValidJulianDate()
             ,

             PreviousValidGregorianDate()
             ,
             NextValidHeisDate()
```

1.95 Date/PreviousValidJulianDate

```
                  NAME
PreviousValidJulianDate -- Returns the prev. valid date (V33.236)

 SYNOPSIS
date_PreviousValidJulianDate(day,month,year,dday,dmonth,dyear);
            d0   d1  d2    a0    a1   a2
```

```
void date_PreviousValidJulianDate(const unsigned short day,
   const unsigned short month, const long year,
   unsigned short *const dday, unsigned short *const dmonth,
   long *const dyear);

void date_PreviousValidJulianDate(const unsigned short day,
   const unsigned short month, const long year,
   unsigned short &dday, unsigned short &dmonth, long &dyear);

 FUNCTION
Returns the previous valid date before a given one.

 INPUTS
day   - Day of the date.
month - Month of the date.
year  - Year of the date.

 RESULT
dday   - Day of the previous valid date
dmonth - Month of the previous valid date
dyear  - Year of the previous valid date

 EXAMPLE
...
date_PreviousValidJulianDate(29,2,2000,&day,&month,&year);
...

 NOTES
None.

 BUGS
None.

 SEE ALSO

              ValidJulianDate()
              ,
              PreviousValidGregorianDate()
              ,

              PreviousValidHeisDate()
              ,
              NextValidJulianDate()
```

1.96 Date/RangeDiff

```
                 NAME
RangeDiff -- Calculates the range between 2 dates. (V33.310)

 SYNOPSIS
date_RangeDiff(day1,month1,year1,day2,month2,year2,days,
    months,years,calendar);
     d0    d1    d2    d3  d4    d5    a0
```

```
      a1     a2    d6

void date_RangeDiff(unsigned short day1,
    unsigned short month1, long year1, unsigned short day2,
    unsigned short month2, long year2, short *const days,
    short *const months, long *const years,
    const date_Calendars calendar);

void date_RangeDiff(unsigned short day1,
    unsigned short month1, long year1, unsigned short day2,
    unsigned short month2, long year2, short &days, short &months,
    long &years, const date_Calendars calendar);

 FUNCTION
RangeDiff gives you back the number of days, months and
years between two specified dates.

 INPUTS
day1     - day of the first date
month1   - month of the first date
year1    - year of the first date
day2     - day of the second date
month2   - month of the second month
year2    - year of the second date
calendar - Calendar system to use

 RESULT
days   - The number of days between the two dates
   (positive if date1 <= date2).
months - The number of months between the two dates
   (positive if date1 <= date2).
years  - The number of years between the two dates
   (positive if date1 <= date2).

 EXAMPLE
short days,months;
long years;
...
date_RangeDiff(18,9,1970,25,1,1998,&days,&months,&years,
    date_Gregorian);
printf("Age of Kai Hofmann is : %ld years, %hd months, %hd days\n",
   years,months,days);
...

 NOTES
It is better only to use this function for years from 8 to 8000!

 BUGS
If you use one of the dates 5.10.1582 to 14.10.1582 you will get an
incorrect output because these days don't exist, except in the
Julian calendar!

 SEE ALSO

              JulianRangeDiff()
              ,
              GregorianRangeDiff()
```

```
               ,
               HeisRangeDiff()
```

1.97 Date/ScaligerYearToG

```
                   NAME
ScaligerYearToG -- Returns the Scaliger year as Gregorian year. (V33)

 SYNOPSIS
year = date_ScaligerYearToG(syear);
 d0           d0

long date_ScaligerYearToG(const unsigned long syear);

 FUNCTION
Returns the Gregorian year of a Scaliger year.

 INPUTS
syear   - Scaliger year

 RESULT
year - The Gregorian year

 EXAMPLE
...
year = date_ScaligerYearToG(6400);
...

 NOTES
It is better to use this function only from 4707 to 7981!

 BUGS
unknown.

 SEE ALSO

               ScaligerYearToJ()
               ,
               ScaligerYearToH()
```

1.98 Date/ScaligerYearToH

```
                   NAME
ScaligerYearToH -- Returns the Scaliger year as Heis year. (V33)

 SYNOPSIS
year = date_ScaligerYearToH(syear);
 d0           d0

long date_ScaligerYearToH(const unsigned long syear);
```

```
 FUNCTION
Returns the Heis year of a Scaliger year.

 INPUTS
syear   - Scaliger year

 RESULT
year - The Heis year

 EXAMPLE
...
year = date_ScaligerYearToH(7000);
...

 NOTES
It is better to use this function only from 4707 to 7981!

 BUGS
unknown.

 SEE ALSO

              ScaligerYearToJ()
              ,
              ScaligerYearToG()
```

1.99 Date/ScaligerYearToJ

```
                  NAME
ScaligerYearToJ -- Returns the Scaliger year as Julian year. (V33)

 SYNOPSIS
year = date_ScaligerYearToJ(syear);
 d0           d0

long date_ScaligerYearToJ(const unsigned long syear);

 FUNCTION
Returns the Julian year of a Scaliger year.

 INPUTS
syear   - Scaliger year

 RESULT
year - The Julian year

 EXAMPLE
...
year = date_ScaligerYearToJ(4800);
...

 NOTES
It is better to use this function only from 4707 to 6295!
```

```
 BUGS
unknown.

 SEE ALSO

              ScaligerYearToG()
              ,
              ScaligerYearToH()
```

1.100 Date/ScaligerYearToYear

```
                  NAME
ScaligerYearToG -- Returns the Scaliger year as  year. (V33.310)

 SYNOPSIS
year = date_ScaligerYearToG(syear,calendar);
 d0           d0    d1

long date_ScaligerYearToG(const unsigned long syear,
    const date_Calendars calendar);

 FUNCTION
Returns the  year of a Scaliger year.

 INPUTS
syear  - Scaliger year
calendar - Calendar system to use

 RESULT
year - The  year

 EXAMPLE
...
year = date_ScaligerYearToG(6400,date_Gregorian);
...

 NOTES
It is better to use this function only from 4707 to 7981!

 BUGS
unknown.

 SEE ALSO

              ScaligerYearToJ()
              ,
              ScaligerYearToG()
              ,
              ScaligerYearToH()
```

1.101 Date/SecToTime

```
                  NAME
SecToTime -- Returns the time from seconds (V33)

 SYNOPSIS
time_SecToTime(secs,hour,min,sec);
    d0   a0  a1  a2

void time_SecToTime(unsigned long secs, unsigned short *const hour,
    unsigned short *const min, unsigned short *const sec);

void time_SecToTime(unsigned long secs, unsigned short &hour,
    unsigned short &min, unsigned short &sec);

 FUNCTION
Gives you back the time from the specified seconds

 INPUTS
secs - Time in seconds

 RESULT
hour - hours (0-23)
min  - minutes (0-59)
sec  - seconds (0-59)

 EXAMPLE
...
time_SecToTime(76860,&hour,&min,&sec);
...

 NOTES
Don't forget to convert 24h time to AM/PM time if needed!

 BUGS
No errorcheck, if you use a valid time

 SEE ALSO

              TimeToSec()
```

1.102 Date/SetCountry

```
                  NAME
SetCountry -- Sets the country defaults for the date lib. (V33.140)

 SYNOPSIS
date_SetCountry(country);
       d0

void date_SetCountry(const date_Countries country);

 FUNCTION
```

```
SetCountry sets the defaults for your country.
Like: correct start of Gregorian calendar.

 INPUTS
country - Country to set.

 RESULT
None.

 EXAMPLE
...
date_SetCountry(unknown);
...

 NOTES
Will *not* work correct if something other than 'unknown' is set
for the moment!

 BUGS
None.

 SEE ALSO

              SetFirstWeekday()
```

1.103 Date/SetFirstWeekday

```
                  NAME
SetFirstWeekday -- Sets the FirstWeekday default (V33.165)

 SYNOPSIS
date_SetFirstWeekday(weekday);
      d0

void date_SetFirstWeekday(const date_Weekdays weekday);

 FUNCTION
SetFirstWeekday sets the weekday with which a week starts.
Default is: Monday

 INPUTS
weekday - Weekday to set.

 RESULT
None.

 EXAMPLE
...
date_SetFirstWeekday(Sunday);
...

 NOTES
dayerr can not be set!
```

```
 BUGS
None.

 SEE ALSO

              SetCountry()
              ,
              JulianWeek()
              ,
              GregorianWeek()
              ,
              HeisWeek()
```

1.104 Date/SupplementCentury

```
 NAME
SupplementCentury -- Add century for a 2 digit year (V33.165)

 SYNOPSIS
year = date_SupplementCentury(year,actualyear);
 d0           d0       d1

long date_SupplementCentury(unsigned short year, long actualyear);

 FUNCTION
Supplements the century for a two digit year by comparing it to the
actual year, which must be a four digit year.
This will be done by using the 'sliding window' technic.

 INPUTS
year     - Two digit year which should be supplemented to four
       digits.
actualyear - The actual year as a four digit year.

 RESULT
year - The year known by a four digit format.

 EXAMPLE
...
printf("%ld\n",SupplementCentury(01,1996));
...

 NOTES
None.

 BUGS
No known bugs.

 SEE ALSO
```

1.105 Date/SYearToJD

```
                  NAME
SYearToJD -- Calcs the JD from a Scaliger year. (V33.310)

 SYNOPSIS
jd = date_SYearToJD(syear,calendar);
d0          d0     d1

unsigned long date_SYearToJD(const unsigned long syear,
    const date_Calendars calendar);

 FUNCTION
Returns the Julianday of a Scaliger year.

 INPUTS
syear  - Scaliger year
calendar - Calendar system to use

 RESULT
jd - The Julianday

 EXAMPLE
...
jd = date_SYearToJD(4800,date_Gregorian);
...

 NOTES
It is better to use this function only from 4707 to 7981!

 BUGS
unknown.

 SEE ALSO

              JSYearToJD()
              ,
              GSYearToJD()
              ,
              HSYearToJD()
```

1.106 Date/TimeDiff

```
                  NAME
TimeDiff -- Returns the difference in seconds (V33)

 SYNOPSIS
secs = time_TimeDiff(hour1,min1,sec1,hour2,min2,sec2);
 d0          d0    d1   d2   d3    d4   d5

long time_TimeDiff(const unsigned short hour1,
    const unsigned short min1, const unsigned short sec1,
    const unsigned short hour2, const unsigned short min2,
```

```
    const unsigned short sec2);

 FUNCTION
Gives you back the difference between the first and the second time
in seconds.

 INPUTS
hour1 - hours of the first time
min1  - minutes of the first time
sec1  - seconds of the first time
hour2 - hours of the second time
min2  - minutes of the second time
sec2  - seconds of the second time

 RESULT
secs - The difference betwen time1 and time1 in seconds.

 EXAMPLE
...
secs = time_TimeDiff(21,15,00,22,0,0);
...

 NOTES
Don't forget to convert AM/PM time to 24h time!
use
              SecToTime()
               to convert the seconds back to a hour,min,secs
format!

 BUGS
No errorcheck, if you use a valid time

 SEE ALSO

              SecToTime()
              ,
              TimeToSec()
```

1.107 Date/TimeToJD

```
                  NAME
TimeToJD -- Returns the JD for a time. (V33)

 SYNOPSIS
jd = time_TimeToJD(hour,min,sec);
d0        d0   d1  d2

float time_TimeToJD(const unsigned short hour,
    const unsigned short min, const unsigned short sec);

 FUNCTION
Returns the JD for a specified time.

 INPUTS
```

```
hour - hour of the time to convert
min  - minute of the time to convert
sec  - sec. of the time to convert

 RESULT
jd - This is the JD time

 EXAMPLE
...
jd = time_TimeToJD(16,33,0);
...

 NOTES
none

 BUGS
There is no check, if the specified time is a valid time!

 SEE ALSO

              JDToTime()
```

1.108 Date/TimeToNum

```
                  NAME
TimeToNum -- Returns the time in a numeric format. (V33.250)

 SYNOPSIS
num = time_TimeToNum(hour,min,sec);
d0          d0   d1  d2

unsigned long time_TimeToNum(const unsigned short hour,
    const unsigned short min, const unsigned short sec);

 FUNCTION
Returns the time in a numeric format.

 INPUTS
hour - hour of the time to convert
min  - minute of the time to convert
sec  - second of the time to convert

 RESULT
num - Numeric time format:
      (hour*100+min)*100+sec

 EXAMPLE
...
num = time_TimeToNum(12,31,0);
...

 NOTES
none
```

```
 BUGS
There is no check, if the specified time is a valid time!

 SEE ALSO

              NumToTime()
```

1.109 Date/TimeToSec

```
                  NAME
TimeToSec -- Returns the time in seconds (V33)

 SYNOPSIS
secs = time_TimeToSec(hour,min,sec);
 d0          d0   d1  d2

unsigned long time_TimeToSec(const unsigned short hour,
    const unsigned short min, const unsigned short sec);

 FUNCTION
Gives you back the time in seconds

 INPUTS
hour - hours you want (0-23)
min  - minutes you want (0-59)
sec  - seconds you want (0-59)

 RESULT
secs - Time in seconds

 EXAMPLE
...
secs = time_TimeToSec(21,15,00);
...

 NOTES
Don't forget to convert AM/PM time to 24h time!

 BUGS
No errorcheck, if you use a valid time

 SEE ALSO

              SecToTime()
```

1.110 Date/TimeZoneFactor

```
 NAME
TimeZoneFactor -- Returns the value you have to add to GMT time (V33)
```

```
 SYNOPSIS
addhours = time_TimeZoneFactor(degrees);
   d0        d0

short time_TimeZoneFactor(const short degree);

 FUNCTION
This gives you the hours you have to add to GMT time,
specified on the fact, that a timezone is 15 degrees
and that GMT is centered on 0 degrees!

 INPUTS
degrees - Position of timezone you live in
(from -180 east to +180 west)

 RESULT
addhours - Time to add to GMT time to get your local time zone
    (-12 to +12)

 EXAMPLE
...
addhours = time_TimeZoneFactor(-8);
...

 NOTES
none

 BUGS
No errorcheck, if you put in valid degrees (-180 to +180)
Only full degrees are supportet, keep sure that you
round in the right way for 0.x degree places
I am not sure about the correct +/- behaviour!!!

 SEE ALSO
```

1.111 Date/ValidDate

```
                NAME
ValidDate -- Checks if the date is a valid date (V33.310)

 SYNOPSIS
valid = date_ValidDate(day,month,year,calendar);
  d0      d0   d1  d2     d3

bool date_ValidDate(const unsigned short day,
    const unsigned short month, const long year,
    const date_Calendars calendar);

 FUNCTION
ValidDate checks if the date is valid.

 INPUTS
day      - Day of the date.
month    - Month of the date.
```

```
year     - Year of the date.
calendar - Calendar system to use

 RESULT
valid - true  : The date is ok.
  false : This is not a correct date!

 EXAMPLE
...
if (date_ValidDate(29,2,2000,date_Gregorian))
  printf("ok\n");
else
  printf("wrong date!!!\n");
...

 NOTES
None.

 BUGS
None.

 SEE ALSO

              ValidTime()
              ,
              ValidJulianDate()
              ,
              ValidGregorianDate()
              ,
              ValidHeisDate()
```

1.112 Date/ValidGregorianDate

```
                  NAME
ValidGregorianDate -- Checks if the date is a valid date (V33.135)

 SYNOPSIS
valid = date_ValidGregorianDate(day,month,year);
  d0        d0   d1  d2

bool date_ValidGregorianDate(const unsigned short day,
    const unsigned short month, const long year);

 FUNCTION
ValidGregorianDate checks if the date is valid.

 INPUTS
day   - Day of the date.
month - Month of the date.
year  - Year of the date.

 RESULT
valid - true  : The date is ok.
  false : This is not a correct date!
```

```
 EXAMPLE
...
if (date_ValidGregorianDate(29,2,2000))
  printf("ok\n");
else
  printf("wrong date!!!\n");
...

 NOTES
None.

 BUGS
None.

 SEE ALSO

              ValidTime()
              ,
              ValidJulianDate()
              ,
              ValidHeisDate()
```

1.113 Date/ValidHeisDate

```
                  NAME
ValidHeisDate -- Checks if the date is a valid date (V33.135)

 SYNOPSIS
valid = date_ValidHeisDate(day,month,year);
  d0          d0 d1  d2

bool date_ValidHeisDate(const unsigned short day,
    const unsigned short month, const long year);

 FUNCTION
ValidHeisDate checks if the date is valid.

 INPUTS
day   - Day of the date.
month - Month of the date.
year  - Year of the date.

 RESULT
valid - true  : The date is ok.
  false : This is not a correct date!

 EXAMPLE
...
if (date_ValidHeisDate(29,2,2000))
  printf("ok\n");
else
  printf("wrong date!!!\n");
...
```

```
 NOTES
None.

 BUGS
None.

 SEE ALSO

              ValidTime()
              ,
              ValidJulianDate()
              ,
              ValidGregorianDate()
```

1.114 Date/ValidJulianDate

```
                  NAME
ValidJulianDate -- Checks if the date is a valid date (V33.135)

 SYNOPSIS
valid = date_ValidJulianDate(day,month,year);
  d0            d0   d1  d2

bool date_ValidJulianDate(const unsigned short day,
    const unsigned short month, const long year);

 FUNCTION
ValidJulianDate checks if the date is valid.

 INPUTS
day   - Day of the date.
month - Month of the date.
year  - Year of the date.

 RESULT
valid - true  : The date is ok.
  false : This is not a correct date!

 EXAMPLE
...
if (date_ValidJulianDate(29,2,2000))
  printf("ok\n");
else
  printf("wrong date!!!\n");
...

 NOTES
None.

 BUGS
None.

 SEE ALSO
```

```
               ValidTime()
               ,
               ValidGregorianDate()
               ,
               ValidHeisDate()
```

1.115 Date/ValidTime

```
                   NAME
ValidTime -- Checks if the time is a valid 24h-format time (V33.135)

 SYNOPSIS
valid = time_ValidTime(hour,min,sec);
  d0      d0   d1  d2

bool time_ValidTime(const unsigned short hour,
    const unsigned short min, const unsigned short sec);

 FUNCTION
ValidTime checks if the time (24h format only) is valid.

 INPUTS
hour - Hour of the time.
min  - Minute of the time.
sec  - Second of the time.

 RESULT
valid - true  : The time is ok.
  false : This is not a correct time!

 EXAMPLE
...
if (time_ValidTime(25,0,0))
  printf("ok\n");
else
  printf("wrong time!!!\n");
...

 NOTES
None.

 BUGS
None.

 SEE ALSO

               ValidJulianDate()
               ,
               ValidGregorianDate()
               ,
               ValidHeisDate()
```

1.116 Date/Week

```
                  NAME
Week -- Gets the weeknumber for a specified date. (V33.310)

 SYNOPSIS
weeknr = date_Week(day,month,year,calendar);
  d0       d0   d1   d2     d3

unsigned short date_Week(const unsigned short day,
    const unsigned short month, const long year,
    const date_Calendars calendar);

 FUNCTION
Week gets the number of the week for a specified date.

 INPUTS
day      - day of the date
month    - month of the date
year     - year of the date
calendar - Calendar system to use

 RESULT
week - This is the number of the week the specified date lies in.
    If the first day in a new year is a Friday, Saturday or
    Sunday, this would be the last week of the last year!
    If the 29.12. is a Monday, the 30.12. is a Monday or a Tuesday,
    the 31.12. is a Monday, Tuesday or a Wednesday this is the
    first week of the next year!

 EXAMPLE
...
weeknr = date_Week(4,10,1582,date_Gregorian);
...

 NOTES
It is better only to use this function for years from 8 to 3200!

 BUGS
For years < 0 errors could occur.

 SEE ALSO

              JulianWeek()
              ,
              GregorianWeek()
              ,
              HeisWeek()
              ,
              SetFirstWeekday()
```

1.117 Date/Weekday

```
                NAME
Weekday -- Gets the weekday of a specified date. (V33.310)

 SYNOPSIS
weekday = date_Weekday(day,month,year,calendar);
  d0      d0   d1   d2    d3

date_Weekdays date_Weekday(const unsigned short day,
    unsigned short month, long year, const date_Calendars calendar);

 FUNCTION
Weekday gets the weekday for a specified date.

 INPUTS
day      - day of the date
month    - month of the date
year     - year of the date
calendar - Calendar system to use

 RESULT
weekday - This result is of type:
    date_Weekdays = {date_dayerr,date_Monday,date_Tuesday,
         date_Wednesday,date_Thursday,date_Friday,
         date_Saturday,date_Sunday};
    dayerr will show you, that an error occurs!

 EXAMPLE
...
weekday = date_Weekday(22,1,1994,date_Gregorian);
if (weekday == dayerr)
 {
  ...
 }
...

 NOTES
It is better only to use this function for years from 8 to 3200!
In this version dayerr will only occur for the lost days :)

 BUGS
It's not possible to use years < 0 (for more see
             JulianWeekday()
             ).

 SEE ALSO

             JulianWeekday()
             ,
             GregorianWeekday()
             ,
             HeisWeekday()
```

1.118 Date/WeekdayShortText

```
                  NAME
WeekdayShortText -- Get the weekday as short text string. (V33.092)

 SYNOPSIS
maxlen = date_WeekdayShortText(wday,wtext,lang);
d0        d0   a0    d1

unsigned short date_WeekdayShortText(const date_Weekdays wday,
    char *const wtext, const date_Languages lang);

 FUNCTION
This function gets the short text string for the weekday-number.

 INPUTS
wday  - Weekday to transform into a string.
wtext - Pointer to a string to fill in the short weekday-text.
lang  - Language for which you want the short weekday-text.

 RESULT
maxlen - Maximum possible length for the weekday-string, this should
    help you if you want to justify the string right or if you
    want to center it! (Normally it's two or three!)
    0 indicates an error!

 EXAMPLE
...
char wtxt[3];
...
maxlen = date_WeekdayShortText(Monday,&wtxt,English);
...

 NOTES
Languages:
Locale    : This is an Amiga >= OS2.1 only feature, for <= OS2.0
      and other systems it will return English text!

 BUGS
In this version there is no check, if there is enough space in
wtext!

 SEE ALSO

              WeekdayText()
              ,
              MonthText()
              ,
              MonthShortText()
```

1.119 Date/WeekdayText

```
                  NAME
```

```
WeekdayText -- Get the weekday as text string. (V33.091)

 SYNOPSIS
maxlen = date_WeekdayText(wday,wtext,lang);
  d0         d0 a0    d1

unsigned short date_WeekdayText(const date_Weekdays wday,
    char *const wtext, const date_Languages lang);

 FUNCTION
This function gets the text string for the weekday-number.

 INPUTS
wday  - Weekday to transform into a string.
wtext - Pointer to a string to fill in the weekday-text.
lang  - Language for which you want the weekday-text.

 RESULT
maxlen - Maximum possible length for the weekday-string, this should
    help you if you want to justify the string right or if you
    want to center it!
    0 indicates an error!

 EXAMPLE
...
char wtxt[20];
...
maxlen = date_WeekdayText(Monday,&wtxt,English);
...

 NOTES
Languages:
Locale    : This is an Amiga >= OS2.1 only feature, for <= OS2.0
      and other systems it will return English text!

 BUGS
In this version there is no check, if there is enough space in
wtext!

 SEE ALSO

              MonthText()
              ,
              WeekdayShortText()
              ,
              MonthShortText()
```

1.120 Date/WWtoDM

```
                  NAME
WWtoDM -- Convert Weekday,Week to Day,Month. (V33.310)

 SYNOPSIS
date_WWtoDM(weekday,week,year,dday,dmonth,dyear,calendar);
```

```
    d0    d1   d2 a0    a1     a2    d3

void date_WWtoDM(const date_Weekdays weekday,
   const unsigned short week, const long year,
   unsigned short *const dday, unsigned short *const dmonth,
   long *const dyear, const date_Calendars calendar);

void date_WWtoDM(const date_Weekdays weekday,
   const unsigned short week, const long year,
   unsigned short &dday, unsigned short &dmonth,
   long &dyear, const date_Calendars calendar);

 FUNCTION
Converts weekday,week to day,month.

 INPUTS
weekday  - weekday of the date to convert
week   - week of the date to convert
year   - year of the date to convert
calendar - Calendar system to use

 RESULT
day   - day of the converted date
month - month of the converted date
year  - year of the converted date

 EXAMPLE
unsigned short day,month;
long year;
...
date_WWtoDM(date_Thursday,14,1997,&day,&month,&year,date_Gregorian);
/* 1997-04-03 */
...

 NOTES
It is better to use this function only from 8 to 8000!
Be careful, the resulting year might be different from the input
year!

 BUGS
unknown.

 SEE ALSO

              JulianWWtoDM()
              ,
              GregorianWWtoDM()
              ,
              HeisWWtoDM()
```

1.121 Date/YearDays

```
                  NAME
YearDays -- Gives back the number of days in a year. (V33.310)
```

```
 SYNOPSIS
days = date_YearDays(year,calendar);
 d0          d0    d1

unsigned short date_YearDays(const long year,
    const date_Calendars calendar);

 FUNCTION
YearDays gives you back the number of days in
a specified year.

 INPUTS
year     - The year in which to count the days.
calendar - Calendar system to use

 RESULT
days - The number of days the year uses.

 EXAMPLE
unsigned short days;
...
days = date_YearDays(1994,date_Gregorian);
printf("Days of 1994 : %hu\n",days);
...

 NOTES
It is better only to use this function for years from 8 to 8000!

 BUGS
No known bugs.

 SEE ALSO

              JulianYearDays()
              ,
              GregorianYearDays()
              ,
              HeisYearDays()
```

1.122 Date/YearToScaliger

```
                  NAME
YearToScaliger -- Returns the year as Scaliger year. (V33.310)

 SYNOPSIS
syear = date_YearToScaliger(year,calendar);
 d0           d0    d1

unsigned long date_YearToScaliger(const long year,
    const date_Calendars calendar);

 FUNCTION
Returns the Scaliger year.
```

```
 INPUTS
year   - year
calendar - Calendar system to use

 RESULT
syear - The Scaliger year

 EXAMPLE
...
syear = date_YearToScaliger(1994,date_Gregorian);
...

 NOTES
It is better to use this function only from 8 to 3200!

 BUGS
unknown.

 SEE ALSO

              JYearToScaliger()
              ,
              GYearToScaliger()
              ,
              HYearToScaliger()
```