Description of calendar procedures from *Calendar.dll* library

Mikołaj Hajduk

E-mail: mikhajduk@gmail.com
Homepage: http://mikhajduk.houa.org

June 16th, 2008

Contents

I. Preface

Creation of the library containing functions for inter-calendar conversions (between Julian and Gregorian calendars) was mainly inspired by my interest to history (especially to its ancient part). In addition it was interesting challenge from the mathematical and programming points of view. First I had to create mathematical models of all calendar functions which were then implemented in FASM assembler. In the first version of this library calendar functions were limited to the time interval starting from the beginning of the common era (for the Julian calendar) and from 15 Oct 1582 (for the Gregorian calendar). Ending point was some day in the future distant 11 million years from the present day. But many dates interesting for historians is placed before beginning of the common era. Consequently, it was necessary to reconstruct mathematical models of the calendar functions and their implementations in such a way that they still could be used in the time interval of 11 million years but with the beginning of the common era placed almost exactly in the middle of this distance. Library functions were extended for dates preceding introduction of the Gregorian calendar and 1 Jan 1 CE for the Julian calendar (such calendars are named *proleptic*). After all these changes every date conversion between both systems is very convenient.

This library has description in three languages (English, Russian and Polish) in order to maximize the number of the potential users.

II. Historical context

During the long evolution of the human civilization and after many astronomical observations length of the solar year was roughly estimated to 365 days. Such year was a base of the Egyptian calendar. More precise observations of moving of the star Sopdet (Sirius, Greek Sothis) made by Egyptian priests led them to the conclusion that average year length is equal to 365.25 days. This approximation was used for building a new calendar during reform introduced by Julius Caesar in 46 BCE. In Julian calendar every fourth (leap) year of the cycle is longer from the others and has 366 days.

However, average length of the Julian year is to big in comparison to the length of the mean solar year which has 365.2422 days. In consequent, after 128 years in the Julian calendar appears additional erroneous day. Julian calendar is ”too slow” and real date of the vernal equinox more and more precede over time date estimated during the reign of the Constantine the Great in 325 to March 21th. Such situation was very inconvenient because in the Christian liturgical calendar date of Easter is calculated in dependence of date of the vernal equinox. Here is a principle for determining the date of Easter:

Easter is the first Sunday after the first full moon that is on or after the ecclesiastical vernal equinox.

In order to solve this problem Pope Gregory XIII introduced the new reform of calendar which was based on the two principles:

- correction of the 10-day difference between ecclesiastical date of the vernal equinox and the real astronomical event which was an effect of the Julian's calendar inaccuracy,
- introduction of the new principle for determining leap years - every year which number is divisible by 4 is leap with exception for the year numbers divisible by 100 and not divisible by 400.

This reformed calendar (named Gregorian since that time) was adopted by most countries in the world, however it was long and complicated process. In the table presented below are shown dates of introduction of the Gregorian calendar in chosen countries:

Date	Country
1582	Italy, Spain, Portugal, Poland and France
1700	Germany
1752	Great Britain
1753	Sweden
1873	Japan
1916	Bulgaria
1917	Turkey
1918	Russia
1919	Romania
1923	Greece
1949	People Republic of China

This table contains chosen dates (in Julian and Gregorian calendars) from the time interval which is accessible for calendar functions from described library:

Day number	Julian date	Gregorian date	Day of week	Historical event
1	1 Jan 5843880 BCE	30 Dec 5844001 BCE	Saturday	Conventional beginning date
2134298452	12 Sep 490 BCE	7 Sep 490 BCE	Thursday	Battle of Marathon
2134356546	1 Oct 331 BCE	26 Oct 331 BCE	Friday	Battle of Gaugamela
2134477171	1 Jan 1	30 Dec 1 BCE	Saturday	Beginning of the Common Era
2134505895	24 Aug 79	22 Aug 79	Tuesday	Explosion of Vesuvius
2135007662	29 May 1453	7 Jun 1453	Tuesday	Fall of Constantinople
2135022043	12 Oct 1492	21 Oct 1492	Friday	Discovery of America
2135054907	4 Oct 1582	14 Oct 1582	Thursday	Control date
2135054908	5 Oct 1582	15 Oct 1582	Friday	Introduction of the Gregorian calendar
2135188665	19 Dec 1948	1 Jan 1949	Saturday	Control date
2135207292	19 Dec 1999	1 Jan 2000	Saturday	Control date
2135210376	29 May 2008	11 Jun 2008	Wednesday	Control date
$2^{32}-1$	3 Aug 5915100	17 Jan 5915222	Monday	Conventional ending date

III. Description of library functions

This section contains description of the calendar functions in the programming context. More detailed mathematical description of these function is presented in the subsection IV.2.

Caution: for dates preceding beginning of the common era we assume that the year number is negative. The year 0 doesn't exist in this numbering system (in contradistinction to the astronomical calendar).

`DWORD DayOfWeek(DWORD Y, DWORD M, DWORD D, DWORD Gregorian)`

Description
This function calculates the day of the week for the given date. Each day of the week is identified by number: 0 - Sunday, 1 - Monday, 2 - Tuesday, 3 - Wednesday, 4 - Thursday, 5 - Friday, 6 - Saturday.

Parameters

- Y - year,
- M - month,
- D - day,
- Gregorian - chosen calendar (0 - Julian, 1 - Gregorian).

Returned values

- $0, 1, \ldots, 6$ if the date is valid,
- -1 for the invalid parameters.

`DWORD IsLeapYear(DWORD Y, DWORD Gregorian)`

Description
This function determines if the given year is leap in the chosen calendar.

Parameters

- Y - year,
- Gregorian - chosen calendar (0 - Julian, 1 - Gregorian).

Returned values

- 1 if the year Y is leap, 0 - in opposite case,
- -1 for the invalid parameters.

`DWORD MDToDayNum(DWORD M, DWORD D, DWORD LeapYearFlag)`

Description
This function calculates the ordinal number of the day in the year.

Parameters

- M - month,
- D - day,
- LeapYearFlag - flag determining if the year is leap (0 - normal year, 1 - leap year).

Returned values

- $1, 2, \ldots, 365$ for the normal year, $1, 2, \ldots, 366$ for the leap year,
- -1 for the invalid parameters.

`DWORD DayNumToMD(DWORD n, DWORD LeapYearFlag, DWORD* M, DWORD* D)`

Description
This function converts the ordinal number of the day in the year to the adequate month and day numbers. The result strongly depends on the flag determining if the year is leap.

Parameters

- n - number of the day in the year,
- LeapYearFlag - flag determining if the year is leap (0 - normal year, 1 - leap year),
- M - pointer to variable where the calculated month number will be stored,
- D - pointer to variable where the calculated day number will be stored.

Returned values

- 0 for the valid parameters $(n, LeapYearFlag)$,
- -1 in opposite case.

```
DWORD DateToAbsDayNum(DWORD Y, DWORD M, DWORD D, DWORD Gregorian)
```

Description
This function calculates the absolute day number for the given date.

Parameters

- Y - year,
- M - month,
- D - day,
- Gregorian - chosen calendar (0 - Julian, 1 - Gregorian).

Returned values

- $1, 2, \ldots, 2^{32} - 1$ for the valid date in the chosen calendar,
- 0 for the invalid parameters.

```
DWORD AbsDayNumToDate(DWORD N, DWORD Gregorian, DWORD* Y, DWORD* M, DWORD* D)
```

Description
This function converts the absolute day number $N \in \{1, 2, \ldots, 2^{32} - 1\}$ to the adequate date (for the chosen calendar).

Parameters

- N - absolute day number,
- Gregorian - chosen calendar (0 - Julian, 1 - Gregorian),
- Y - pointer to variable where the calculated year number will be stored,
- M - pointer to variable where the calculated month number will be stored,

- D - pointer to variable where the calculated day number will be stored.

Returned values

- 0 for the valid parameters $(N, Gregorian)$,
- -1 in opposite case.

```
DWORD GregorianToJulian(DWORD Yg, DWORD Mg, DWORD Dg, DWORD* Yj, DWORD* Mj, DWORD* Dj)
```

Description
This function converts the Gregorian date to the adequate Julian date.

Parameters

- Yg - year of the Gregorian date,
- Mg - month of the Gregorian date,
- Dg - day of the Gregorian date,
- Yj - pointer to variable where the calculated year number of the Julian date will be stored,
- Mj - pointer to variable where the calculated month number of the Julian date will be stored,
- Dj - pointer to variable where the calculated day number of the Julian date will be stored.

Returned values

- 0 for the valid Gregorian date,
- -1 in opposite case.

```
DWORD JulianToGregorian(DWORD Yj, DWORD Mj, DWORD Dj, DWORD* Yg, DWORD* Mg, DWORD* Dg)
```

Description
This function converts the Julian date to the adequate Gregorian date.

Parameters

- Yj - year of the Julian date,

- Mj - month of the Julian date,
- Dj - day of the Julian date,
- Yg - pointer to variable where the calculated year number of the Gregorian date will be stored,
- Mg - pointer to variable where the calculated month number of the Gregorian date will be stored,
- Dg - pointer to variable where the calculated day number of the Gregorian date will be stored.

Returned values

- 0 for the valid Julian date,
- -1 in opposite case.

IV. Mathematical model

IV.1. Notational conventions

The symbol $\mathbb{Z}$ denotes the set of integer numbers, $\mathbb{R}$ denotes set of real numbers.

The function $E(x) : \mathbb{R} \to \mathbb{Z}$ (also called *entier* or *floor*) for every real number x returns highest integer number less than or equal to x:

$$E(x) = \lfloor x \rfloor = \max\{k \in \mathbb{Z};\ k \leq x\}$$

For every predicate $\phi(x_1, x_2, \ldots, x_n)$ defined in the set X by the symbol $[\phi(a_1, a_2, \ldots, a_n)]$, where $(a_1, a_2, \ldots, a_n) \in X$, we denote number value equal to 0 or 1 according to the Boolean value of the sentence $\phi(a_1, a_2, \ldots, a_n)$:

$$[\phi(a_1, a_2, \ldots, a_n)] = \begin{cases} 0 & ;\ \phi(a_1, a_2, \ldots, a_n) \text{ is a false sentence} \\ 1 & ;\ \phi(a_1, a_2, \ldots, a_n) \text{ is a true sentence} \end{cases}$$

The constants $C_1, C_4, C_{100}, C_{400}$ are equal to the lengths of base cycles in the Julian and Gregorian calendars:

$C_1 = 365$,	number of days in a normal year,
$C_4 = 4C_1 + 1 = 1461 = 3 * 487$,	number of days in the 4-year cycle (base cycle of the Julian calendar),
$C_{100} = 25C_4 - 1 = 36524$,	number of days in a "normal" century in the Gregorian calendar (i.e. century ending with a normal, 365-day, year),
$C_{400} = 4C_{100} + 1 = 146097 = 3^3 * 7 * 773$,	number of days in the complete 400-year cycle of the Gregorian calendar.

The constant T (which could be named "Great Cycle") is the least common multiple of lengths in days of the Julian 4-year cycle and Gregorian 400-year cycle:

$$T = \mathrm{lcm}(C_4, C_{400}) = 3^3 * 7 * 487 * 773 = 71149239$$

The constants J and G are equal to the numbers of the complete years of the Julian and Gregorian calendars respectively contained in the time interval given by "Great Cycle" T:

$$J = 4E\left(\frac{T}{C_4}\right) = 194796$$

$$G = 400E\left(\frac{T}{C_{400}}\right) = 194800$$

The starting point of the time interval in which calendar functions can work is a day preceding beginning of the common era (i.e. 1 Jan 1 in the Julian calendar) by kT days where

$$k = 30$$

That way beginning of the common era is placed almost in the middle of the time interval supported by calendar functions.

By the symbol $DaySum(M, F)$ we denote the sum of lengths of months preceding given month M, where F means the flag equal to 1 if the year is leap and 0 in opposite case:

$$DaySum : \{1, 2, \ldots, 12\} \times \{0, 1\} \rightarrow \{0, 31, 59, 60, 90, 91, 120, 121, 151, 152, 181, 182, 212, 213, 243, 244, 273, 274, 304, 305, 334, 335\}$$

$$DaySum(M, F) = \sum_{i=12F}^{M-2+12F} MonthLen_i$$

Table presented below contains values of the function $DaySum$ for every pair (M, F) from its domain:

$F \backslash M$	1	2	3	4	5	6	7	8	9	10	11	12
0	0	31	59	90	120	151	181	212	243	273	304	334
1	0	31	60	91	121	152	182	213	244	274	305	335

The symbol $(MonthLen_k)$ denotes finite sequence of numbers which first 12 elements are equal to lengths of months of a normal year while next 12 values are equal to the lengths of months of a leap year:

$$MonthLen : \{0, 1, \ldots, 23\} \rightarrow \{1, \ldots, 31\}$$

$$(MonthLen_k) = (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)$$

IV.2. Mathematical model of calendar functions

This subsection contains description of mathematical models of the calendar functions (implementation of these functions was described in subsection III.).

$DayOfWeek : \mathbb{Z}^4 \rightarrow \{-1, 0, 1, \ldots, 6\}$

This function calculates the day of the week for the given date.

$$DayOfWeek(Y, M, D, Gregorian) = \begin{cases} (N+5) \bmod 7 & ; \ N \in \mathbb{Z} - \{0\} \\ \\ -1 & ; \ N = 0 \end{cases}$$

where N is the absolute number of the day corresponding to the given date.

$$N = DateToAbsDayNum(Y, M, D, Gregorian)$$

$IsLeapYear : \mathbb{Z}^2 \rightarrow \{-1, 0, 1\}$

This function determines if the given year is leap in the chosen calendar.

$$IsLeapYear(Y, Gregorian) = \begin{cases} -1 & ; \ Y = 0 \ \vee \ Gregorian \notin \{0,1\} \\ \\ [Y' \bmod 4 = 0] & ; \begin{cases} Y \neq 0 \ \wedge \ Gregorian = 0 \\ \\ Y \neq 0 \ \wedge \ Gregorian = 1 \ \wedge \\ Y' \bmod 100 \neq 0 \end{cases} \\ \\ \left[E\left(\frac{Y'}{100}\right) \bmod 4 = 0\right] & ; \ \begin{matrix} Y \neq 0 \ \wedge \ Gregorian = 1 \ \wedge \\ Y' \bmod 100 = 0 \end{matrix} \end{cases}$$

where by the symbol Y' we denote modified year number defined as follows:

$$Y' = |Y| - [Y < 0]$$

$MDToDayNum : \mathbb{Z}^3 \to \{-1, 1, \ldots, 366\}$

This function calculates the ordinal number of the day in the year (regarding the fact if the year is leap or not).

$$MDToDayNum(M, D, F) = \begin{cases} DaySum(M, F) + D & ; \begin{cases} M \in \{1, \ldots, 12\} \ \wedge \ F \in \{0, 1\} \ \wedge \\ D \in \{1, \ldots, MonthLen_{12F+M-1}\} \end{cases} \\ \\ -1 & ; \text{ in opposite case} \end{cases}$$

$DayNumToMD : \mathbb{Z}^4 \to \{-1, 0\} \times \mathbb{Z}^2$

This function converts the ordinal number of the day in the year to the adequate month and day numbers.

$$DayNumToMD(n, F, M, D) = \begin{cases} (0, m, n - DaySum(m, F)) & ; \ n \in \{1, \ldots, 366\} \ \wedge \ F \in \{0, 1\} \\ \\ (-1, M, D) & ; \text{ in opposite case} \end{cases}$$

where m is the month number calculated from the ordinal number of the day in the year (regarding value of the leap year flag F):

$$m = \max\{i; \ i \in \{1, \ldots, 12\} \ \wedge \ DaySum(i, F) < n\}$$

$DateToAbsDayNum : \mathbb{Z}^4 \to \mathbb{Z}$

This function calculates the absolute day number for the given date.

$$DateToAbsDayNum(Y, M, D, Gregorian) = \begin{cases} n - 364 & ; \ Y' = 0 \ \wedge \ Y \neq 0 \ \wedge \ Gregorian \in \{0, 1\} \ \wedge \ n \neq -1 \\ \\ N & ; \ Y' \neq 0 \ \wedge \ Y \neq 0 \ \wedge \ Gregorian \in \{0, 1\} \ \wedge \ n \neq -1 \\ \\ 0 & ; \ Y = 0 \ \vee \ Gregorian \notin \{0, 1\} \ \vee \ n = -1 \end{cases}$$

where n denotes the ordinal number of the day in the year Y:

$$n = MDToDayNum(M, D, IsLeapYear(Y, Gregorian))$$

By the symbol Y' we denote the number of the given year calculated from the starting point preceding by kT days beginning of the common era (i.e. 1 Jan 1 in the Julian calendar):

$$Y' = Y + [Y < 0] + kJ + k(G - J)[Gregorian = 1]$$

By the symbol N we denote the day number corresponding to the given date counted from the starting point:

$$N = 365(Y' - 1) + E\left(\frac{Y' - 1}{4}\right) + [Gregorian = 1]\left(E\left(\frac{Y' - 1}{400}\right) - E\left(\frac{Y' - 1}{100}\right) + 2\right) + n$$

$AbsDayNumToDate : \mathbb{Z}^5 \to \{-1, 0\} \times \mathbb{Z}^3$

This function converts the absolute day number $N \in \mathbb{Z}$ to the adequate date (for the chosen calendar).

$$AbsDayNumToDate(N, Gregorian, Y, M, D) = \begin{cases} (-1, Y, M, D) & ;\ N = 0 \ \vee \ Gregorian \notin \{0, 1\} \\ (0, -kG - 1, 12, 29 + N) & ;\ N \in \{1, 2\} \ \wedge \ Gregorian = 1 \\ (0, Y' - [Y' \leq 0], M', D') & ;\ \text{in other cases} \end{cases}$$

The values Y', M' and D' are obtained from the formulas:

$$Y' = Y^* - kJ - k(G - J)[Gregorian = 1]$$

$$(0, M', D') = DayNumToMD(N' + 1,\ IsLeapYear(Y^*, Gregorian), M, D)$$

where the values N' and Y^* are calculated as follows:

$$(N', Y^*) = Q\left(N_{100} \bmod C_4,\ Y_{100} + 4E\left(\frac{N_{100}}{C_4}\right)\right)$$

where the function $Q : \{0, \ldots, C_4\} \times \mathbb{Z} \to \{0, \ldots, C_1 + 1\} \times \mathbb{Z}$ given by the formula

$$Q(x, y) = \left(x - C_1 \min\left(E\left(\frac{x}{C_1}\right), 3\right),\ y + 1 + \min\left(E\left(\frac{x}{C_1}\right), 3\right)\right)$$

converts a pair *(the number of the day in the 4-year cycle, the year number)* into a pair *(the number of the day in the year, the updated year number).*

The values N_{100} and Y_{100} are obtained from the formula:

$$(N_{100}, Y_{100}) = \begin{cases} (N-1,\ 0) & ;\ Gregorian = 0 \\ P\left((N-3) \bmod C_{400},\ 400E\left(\frac{N-3}{C_{400}}\right)\right) & ;\ Gregorian = 1 \end{cases}$$

where the function $P : \{0, \ldots, C_{400}\} \times \mathbb{Z} \to \{0, \ldots, C_{100} + 1\} \times \mathbb{Z}$ given by the formula

$$P(x, y) = \left(x - C_{100} \min\left(E\left(\frac{x}{C_{100}}\right),\ 3\right),\ y + 100 \min\left(E\left(\frac{x}{C_{100}}\right),\ 3\right)\right)$$

converts a pair *(the number of the day in the 400-year cycle, the year number)* into a pair *(the number of the day in the century, the updated year number)*.

$GregorianToJulian : \mathbb{Z}^6 \to \{-1, 0\} \times \mathbb{Z}^3$

This function converts the Gregorian date to the adequate Julian date.

$$GregorianToJulian(Y_g, M_g, D_g, Y_j, M_j, D_j) = AbsDayNumToDate(DateToAbsDayNum(Y_g, M_g, D_g, 1), 0, Y_j, M_j, D_j)$$

$JulianToGregorian : \mathbb{Z}^6 \to \{-1, 0\} \times \mathbb{Z}^3$

This function converts the Julian date to the adequate Gregorian date.

$$JulianToGregorian(Y_j, M_j, D_j, Y_g, M_g, D_g) = AbsDayNumToDate(DateToAbsDayNum(Y_j, M_j, D_j, 0), 1, Y_g, M_g, D_g)$$