Portable Systems Group

NT OS/2 Time Conversion Specification

Author: *Gary D. Kimura*

Revision 1.2, August 14, 1990

1. Introduction 1

2. Converting From TIME to TIME_FIELDS 2

3. Converting From TIME_FIELDS to TIME 2

4. Converting From TIME to OS/2-Based Time 3

5. Converting From OS/2-Based Time to TIME 3

6. Converting From TIME to POSIX-Based Time 4

7. Converting From POSIX-Based Time to TIME 4

1. Introduction

This specification describes the **NT OS/2** routines that implement absolute time conversion. Time, in **NT OS/2**, is expressed as a 64-bit signed large integer value with 100ns resolution and contained in variables of type **TIME**. Absolute times, such as August 29, 1989 at 11:08:30.245, are expressed as positive values (excluding zero), while time intervals (e.g., 1 hour and 20 minutes) are expressed as negative values (including zero).

The routines described in this document deal with absolute time and converting from a value of type **TIME** to a structure representing year, month, day, hour, minute, second, and millisecond, and back again. They do not address the manipulation of time intervals or time zones. Time interval manipulation is simply done using large integer arithmetic, and the time zone issue is beyond the realm of the package.

Though **NT OS/2** time has a 100ns resolution, the routines described here have only a one millisecond resolution. Time precision smaller than a millisecond is either ignored or lost by these routines.

The basis for **NT OS/2** time is the beginning of the year 1601 AD, which is chosen because 1601 AD is the start of a new quadricentury. The algorithms used in this package are based on the Gregorian calendar.

The structure **TIME_FIELDS** is used to represent a time value divided into its logical components. The declaration for **TIME_FIELDS** is the following:

```
typedef struct _TIME_FIELDS {
        CSHORT Year;
        CSHORT Month;
        CSHORT Day;
        CSHORT Hour;
        CSHORT Minute;
        CSHORT Second;
        CSHORT Milliseconds;
        CSHORT Weekday;
} TIME_FIELDS;
```

where:

Year - Denotes the year in a range between 1601 AD to around 30000 AD.

Month - Denotes the month in a range between 1 (January) to 12 (December).

Day - Denotes the day in the month with a range between 1 to either 28, 29, 30, or 31, depending on the month and the year.

Hour - Denotes the hour, on a 24 hour clock, in a range between 0 and 23.

Minute - Denotes the minute in a range between 0 and 59.

Second - Denotes the second in a range between 0 to either 59 or 60, where 60 denotes a leap second.

Milliseconds - Denotes the fraction of a second in a range between 0 and 999.

Weekday - Denotes the day of the week represented by the rest of the time fields in a range between 0 (Sunday) and 6 (Saturday). This field is only used when translating a 64-bit time value into a time field structure and not when mapping the other direction.

In addition to providing routines to convert between **TIME** and a **TIME_FIELDS** structure, this package also provides routines that convert between **TIME** and **OS/2** time and **POSIX** time. **OS/2** time is expressed as the number of seconds since the start of 1980. **POSIX** time is the number of seconds since the start of 1970.

The **APIs** that implement time conversion are the following:

RtlTimeToTimeFields - Converts a **TIME** value to a **TIME_FIELDS** structure.
RtlTimeFieldsToTime - Converts a **TIME_FIELDS** structure to a **TIME** value.
RtlTimeToSecondsSince1980 - Converts a **TIME** value to seconds with a 1980 base.
RtlSecondsSince1980ToTime - Converts seconds with a 1980 base to a **TIME** value.
RtlTimeToSecondsSince1970 - Converts a **TIME** value to seconds with a 1970 base.
RtlSecondsSince1970ToTime - Converts seconds with a 1970 base to a **TIME** value.

2. Converting From TIME to TIME_FIELDS

A **TIME** value is converted to a corresponding **TIME_FIELDS** structure with the **RtlTimeToTimeFields** procedure.

VOID
RtlTimeToTimeFields (
    IN PTIME *Time*,
    OUT PTIME_FIELDS *TimeFields*
    );

Parameters:

Time - Supplies the value being converted

TimeFields - A pointer to the variable being set

The input time can be any non-negative large integer value and is interpreted as the number of 100ns tics since the start of 1601 AD. The resulting TimeFields variable will never contain a leap second value of 60.

3. Converting From TIME_FIELDS to TIME

A **TIME_FIELDS** structure is converted to a corresponding **TIME** value with the **RtlTimeFieldsToTime** procedure.

BOOLEAN
RtlTimeFieldsToTime (
    IN PTIME_FIELDS *TimeFields*,
    OUT PTIME *Time*
    );

Parameters:

TimeFields - Supplies the time field structure initialized by the caller to convert to a time value

Time - A pointer to the variable being set

The function result is TRUE if the input time fields is well formed and is expressible by a time variable and FALSE otherwise.

The input time must be well formed (i.e., the year must be 1601 or later, month must be between 1 and 12, day must be between 1 and the maximum day for the given month and year, hour must be between 0 and 23, minute must be between 0 and 59, second must be between 0 and 60 where the value 60 is only allowed during the last time in a month, and milliseconds must be between 0 and 999). The Weekday field is ignored by this procedure.

4. Converting From TIME to OS/2-Based Time

A **TIME** value is converted to the corresponding number of seconds since the start of 1980 with the **RtlTimeToSecondsSince1980** procedure.

BOOLEAN
RtlTimeToSecondsSince1980 (
    IN PTIME *Time*,
    OUT PULONG *ElapsedSeconds*
    );

Parameters:

Time - Supplies the value being converted, it must represent a time between 1980 AD and around 2115 AD

ElapsedSeconds - A pointer to the variable being set

The function result is TRUE if the input value is within the range expressible by the output value [1980 to 2115] and otherwise FALSE.

5. Converting From OS/2-Based Time to TIME

A **ULONG** value representing the number of elapsed seconds since the start of 1980 is converted to a corresponding **TIME** value with the **RtlSecondsSince1980ToTime** procedure.

VOID
RtlSecondsSince1980ToTime (
    IN ULONG *ElapsedSeconds*,
    OUT PTIME *Time*
    );

Parameters:

ElapsedSeconds - Supplies the value (i.e., number of seconds since the start of 1980) being converted

Time - A pointer to the variable being set

6. Converting From TIME to POSIX-Based Time

A **TIME** value is converted to the corresponding number of seconds since the start of 1970 with the **RtlTimeToSecondsSince1970** procedure.

BOOLEAN
RtlTimeToSecondsSince1970 (
    IN PTIME *Time*,
    OUT PULONG *ElapsedSeconds*
    );

Parameters:

Time - Supplies the value being converted, it must represent a time between 1970 AD and around 2105 AD

ElapsedSeconds - A pointer to the variable being set

The function result is TRUE if the input value is within the range expressible by the output value [1970 to 2105] and otherwise FALSE.

7. Converting From POSIX-Based Time to TIME

A **ULONG** value representing the number of elapsed seconds since the start of 1970 is converted to a corresponding **TIME** value with the **RtlSecondsSince1970ToTime** procedure.

VOID
RtlSecondsSince1970ToTime (
    IN ULONG *ElapsedSeconds*,
    OUT PTIME *Time*
    );

Parameters:

ElapsedSeconds - Supplies the value (i.e., number of seconds since the start of 1970) being converted

Time - A pointer to the variable being set

Revision History:

Original Draft 1.0, August 29, 1989

Revision 1.1, January 4, 1990

1. Included zero time as an interval time.
2. Make 60 a valid value in the second field of the TIME_FIELDS structure to handle leap seconds.

Revision 1.2, August 14, 1990

1. Fix procedure prototype for RtlTimeFieldsToTime to return a BOOLEAN result.