librtc Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

1 Library Overview

Overview

librtc is a library for providing time information. It provides the current time, which can always obtain CPU internal timer managed by the kernel safely.

librtc also provides conversion functions for various types of time formats.

Files

The following files are required to use librtc.

Filename	Description
rtc.h	Header file
libSceRtc_stub.a	Stub library file

2 Usage Procedure

Referencing the Current Time

Since the base time is calculated when librtc is initialized, the current time calculated by librtc can be obtained thereafter without referencing the hardware RTC.

```
SceDateTime clk;

sceRtcGetCurrentClockLocalTime(&clk);      // Get current time (local time)
printf("%04d/%02d/%02d %02d:%02d:%02d\n",
              clk.year, clk.month, clk.day,
              clk.hour, clk.minute, clk.second);
```

If `sceRtcGetCurrentClock()` is used instead of `sceRtcGetCurrentClockLocalTime()`, the local time corresponding to an arbitrary time zone can be obtained. The current time according to UTC time can be obtained by specifying 0 for the time zone offset in `sceRtcGetCurrentClock()`.

Time Information Formatting

librtc also provides processing for converting time information to a string.

```
SceRtcTick tick;
char szBuf[128];

sceRtcGetCurrentTick(&tick);            // Get UTC time
sceRtcFormatRFC2822LocalTime(szBuf, &tick);      // Convert to string with
                                                 // RFC2822 format
printf("Date: %s\n", szBuf);
```

This example first uses `sceRtcGetCurrentTick()` to get the Tick representation of the current time (UTC) and then converts that time to the local time represented in RFC2822 format.

librtc also provides a formatting function for converting to RFC3339 (ISO8601) format.

Referencing Network Time

You can use the `sceRtcGetCurrentNetworkTick()` function to obtain the network time that was synchronized with the time reported by the server at PSN℠ sign-in. Once synchronized, you can continue to obtain the network time even if the network is disconnected or the power is turned off. This state remains in effect until time information is lost when the battery runs out.

When writing specifications that use these functions, be careful not to create any problems for the user.

Referencing Cumulative Time

In contrast to the clock which the user sets via the PlayStation®Vita system software, when the `sceRtcGetAccumulativeTime()` function is used, it is possible to obtain the cumulative time which has been continuously tracked, provided that the battery has not run out.

In addition, by using this function in conjunction with the `sceRtcGetLastReincarnatedTick()` and `sceRtcGetLastAdjustedTick()` functions, it is possible to detect whether or not the clock has been manipulated by the user when the application was not running.

When drafting specifications which utilize these functions, be careful that they do not become detrimental to the user.

3 Precautions

Tick Units

librtc performs operations with 1 microsecond as the smallest unit of time. This value is called a Tick unit, and the time represented by `SceRtcTick` indicates the cumulative number of Ticks from 0001/01/01 00:00:00.

Tick Calculations

Since `SceDateTime` in which fields are separated into elements for year, month, day, hours, minutes, seconds, and milliseconds, is not suited for calculations, librtc performs almost all calculations in terms of Tick units. Calculations using Ticks are provided as the `sceRtcTickAdd*()` functions, which take a pointer to `SceRtcTick` as an argument.

SceRtcTick Range

Since, as described earlier, the value in `SceRtcTick` is represented as a 64-bit value for the number of 1 MHz ticks with the year 0001 as the base point, increasing values starting from 0001/01/01 00:00:00 constitute the range.

MS-DOS-Compatible Format Range

Since, in MS-DOS-compatible time information, the year is represented as a 7-bit value with 1980 as the base point, the values 1980/01/01 00:00:00 to 2107/12/31 23:59:58 constitute the range. For seconds, odd numbers cannot be represented.

Win32 FILETIME-Compatible Format Range

Since Win32 FILETIME-compatible time information is represented as a 64-bit value for the number of 100 nsec ticks with the year 1601 as the base point, the values increasing from 1601/01/01 00:00:00 constitute the range.

POSIX time_t-Compatible Format Range

Since POSIX time_t-compatible time information is represented in one-second ticks as a signed 32-bit value with the year 1970 as the base point, the values 1970/01/01 00:00:00 to 2038/01/19 03:14:07 constitute the range.

`time()` Function in libc

The `time()` function which uses the `time_t` type in libc will reach its maximum at 03:14:07 on January 19, 2038; it has a so-called year 2038 problem. In the system software, the time can be set beyond the year 2038, so care must be exercised when using the `time()` function in libc.

Years 1999 and 2100

Note that in system software, January 1, 2000 through December 31, 2099 can be set, but due to time zone relationships, December 31, 1999 and January 1, 2100 may exist as well.

Relationship between the RTC Service and the Time Management Feature of the Thread Manager

The RTC service uses the RTC hardware together with the time management feature of the thread manager to provide the current time to an application. When the system is booted or resumed from sleep mode, the RTC service is synchronized with the RTC hardware. During normal operation, the current time is calculated using the time management feature of the thread manager without accessing the RTC hardware.

However, during a suspend/resume, there will be a loss of continuity if the time is obtained from the `sceRtcGetCurrentTick()`, `sceRtcGetCurrentClock()`, `sceRtcGetCurrentClockLocalTime()`, or `sceRtcGetCurrentNetworkTick()` functions. The phenomenon of time moving backward may also be observed due to synchronization with the hardware RTC.

Also, when a game is suspended and restarted, the system returns to a state in which the network time obtained by `sceRtcGetCurrentNetworkTick()` is temporarily disabled.

Because of this, do not use `sceRtcGetCurrentTick()` to measure execution time or to perform thread scheduling. The RTC service provides a real-time clock that is linked with real-world time. When a counter is needed during runtime that monotonically increases uniformly, the time management functions provided by the thread manager should be used instead. Each should be used properly according to the desired function.