What clock do MSG.time and GetMessageTime use?

devblogs.microsoft.com/oldnewthing/20140122-00

January 22, 2014

Raymond Chen

The `MSG` structure has a field called `time` which is a `DWORD`. There is also a function `GetMessageTime` which returns a `LONG`. Both are documented as returning the time the message was generated, but the types are different. Are these time units comparable?

Yes, they are the same thing. They all use the 32-bit timer provided by the `GetTickCount` function. Sorry about the inconsistency in signed/unsigned-ness. Feel free to cast between them; they are fundamentally the same thing.

Whether you prefer the signed or unsigned version depends on what you intend to do with the calculation, specifically, how you want to treat the case where the events occurred out of the expected order.

If you want to see whether some amount of time has elapsed, then you probably want to cast the return value of `GetMessageTime` to `DWORD`:

```
if ((DWORD)GetMessageTime() - dwStartTime < VALUE) {
 ...
}
```

The unsigned test checks that the message time is *on or after* the start time, but not more than `VALUE` milliseconds after it. If the message time comes *before* the start time, then the test fails.

On the other hand, if you want to see which of two events occurred earlier, then you want to use a signed subtraction:

```
if (GetMessageTime() - (LONG)dwStartTime < VALUE) {
 ...
}
```

The signed test checks that the message time is *no later than* `VALUE` milliseconds after the start time.

Note that language lawyers may note that the above calculation relies on two’s-complement overflow, and signed arithmetic overflow is specifically called out by the C language as *undefined behavior*. Therefore, a stricter language-compliant version of the above test would be

```
if ((LONG)((DWORD)GetMessageTime() - dwStartTime) < VALUE) {
 ...
}
```

The above version still relies on twos-complement arithmetic, but the Win32 ABI requires twos-complement arithmetic, so we’re safe there at least.

Raymond Chen

Follow