

Re: WaitForSingleObject() will not deadlock

## Re: WaitForSingleObject() will not deadlock

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2007-07/msg00515.html>

---

- *From:* Joseph M. Newcomer <[newcomer@xxxxxxxxxxxxx](mailto:newcomer@xxxxxxxxxxxxx)>
  - *Date:* Fri, 06 Jul 2007 00:27:21 -0400
- 

But what is TSO? According to wikipedia

TSO is a three letter abbreviation commonly used to represent:

Science:

TCP segmentation offloading

Technical Standard Order, a minimum performance standard issued by the FAA for certain items used on civil aircraft.

Time Sharing Option, an interactive command line interpreter for IBM mainframe operating systems MVS/ESA, OS/390 and z/OS

Transmission System Operator

Other:

Tasmanian Symphony Orchestra

Tax Service Office

Territory Sales Organization

The Sacred Octagon, the bi-monthly journal of the New England MG T Register (an organisation of MG enthusiasts)

The Sims Online, a massively multiplayer game.

The Singular Online, a massively multiplayer RPG game.

The Special One, a nickname of Chelsea Manager José Mourinho

The Stationery Office, publishers

Tokyo Symphony Orchestra

Toronto Symphony Orchestra

Trans-Siberian Orchestra

The ICAO code for Transaero

Transportation security officer

A local Sorority at Knox College, Galesburg, IL

Total Shares Outstanding, a financial term referring to the stock held by investors, including restricted shares, as well as those held by the public

Tourist Standard (or Second) Open, a type of British Railways coach.

None of these seem to apply.

\*\*\*\*\*

On Thu, 05 Jul 2007 17:04:34 -0700, Frank Cusack <[fcusack@xxxxxxxxxxxxx](mailto:fcusack@xxxxxxxxxxxxx)> wrote:

On Tue, 03 Jul 2007 13:53:09 -0400 Joseph M. Newcomer <[newcomer@xxxxxxxxxxxxx](mailto:newcomer@xxxxxxxxxxxxx)>

Re: WaitForSingleObject() will not deadlock

## Re: WaitForSingleObject() will not deadlock

wrote:

It seems that "memory visibility" has something to do with making sure that results are in memory. I would have thought this would have been more commonly referred to as the "cache coherency" problem.

They are 2 different problems, but because x86 is TSO, they are equivalent for that platform.

Because of the Interlocked operation used on the x86 and x64, all results in caches and pipes are guaranteed to be in a consistent state throughout the entire multiprocessor cache structure. Therefore, once you have acquired any locking object, you are guaranteed that all processors are seeing the same memory contents, one way or another.

That is true for all processors, but this is because acquisition of a lock is guaranteed to issue a membar, not because of cache coherency.

\*\*\*\*\*

This is particularly an issue because of the write-combining cache of the newer Pentium models.

\*\*\*\*\*

Strictly speaking, shared variables should be declared 'volatile' to ensure the compiler has not kept a copy around, and this is

Strictly speaking, this is not correct. But the MS VC definition of volatile is different than the ISO C definition. This has been discussed to death on [comp.programming.threads](#). Again and again.

completely orthogonal to the concept of locking; it appears that the Microsoft C/C++ compiler is very conservative about code motions across functions and therefore seems less subject to this kind of error, although the formal requirements of the C/C++ language would dictate that the use of 'volatile' is mandatory to ensure the compiler has not done something bad.

Not correct. volatile (the ISO C definition) does no such thing. A mutex is sufficient to guarantee visibility, and the compiler is guaranteed not to reorder code around locking operations.

Re: WaitForSingleObject() will not deadlock

\*\*\*\*

There's a good discussion of these issues in

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1777.pdf>

\*\*\*\*\*

-frank

Joseph M. Newcomer [MVP]

email: [newcomer@xxxxxxxxxxxx](mailto:newcomer@xxxxxxxxxxxx)

Web: <http://www.flounder.com>

MVP Tips: [http://www.flounder.com/mvp\\_tips.htm](http://www.flounder.com/mvp_tips.htm)

.