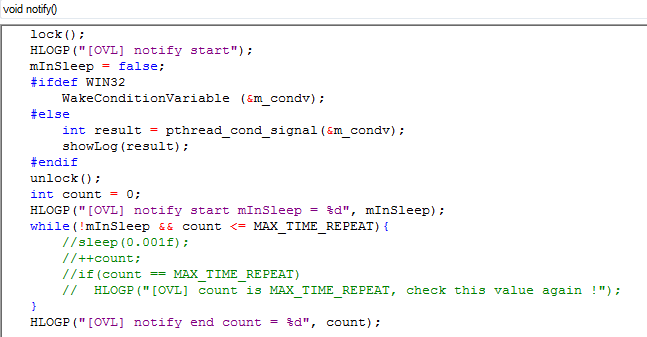
**C++ VOLATILE in Overlay implementation**

**1 – Situation**

Overlay again (please take a glance at C++ CLOCK\_MONOTONIC for simple overlay’s description).

The problem occurred at function ThreadLocker::notify()



The while loop is break if mInSleep = true. In fact, there were some bugs that sometimes it looped infinite even when mInSleep was set to true. Log showed as below:

|  |
| --- |
| [2102-07-03][12:07:31:746] [AppMain] [Model][Thread:3]void ThreadLocker::notify():72 [OVL] notify start  [2102-07-03][12:07:31:746] [AppMain] [Model][Thread:3]void ThreadLocker::notify():82 [OVL] notify start mInSleep = 0 // PC went to while loop  [2102-07-03][12:07:31:747] [AppMain] [Model][Thread:12]int ThreadLocker::wait(uint32\_t):102 [OVL] toan.tv set **mInSleep: 1** // a thread change the value of mInSleep to “true” => it should break the while loop => in fact it doesn’t  [2102-07-03][12:07:31:747] [AppMain] [Model][Thread:12]int ThreadLocker::wait(uint32\_t):104 [OVL] Sleep infinite until having signal to wake up |

If we tried to enable the command sleep(0.001f) in while, it worked normal. But sometimes, issues still happened while booting.

**2. Analysis**

Compiler optimization.

<https://stackoverflow.com/questions/4437527/why-do-we-use-volatile-keyword-in-c>

Consider when above code gets compiled, the compiler may optimize this code, if it finds that the program never ever makes any attempt to change the value of mInSleep. So it may be tempted to optimize the while loop by changing it from

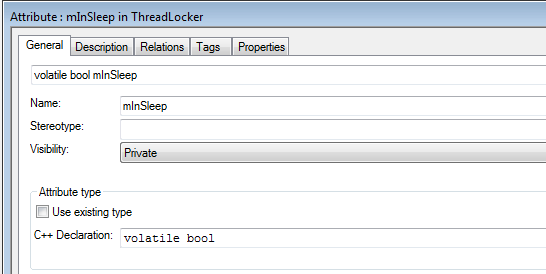
While (**!mInSleep** && …)

To simply

While (**true** && …)

So that the execution could be fast.

Volatile prevents compiler from optimizing code.



Note: change *sleep(0.001f)* to *delay(2)* would get better performance.