Subject: Re: Use same variable in different threads
Posted by Didier on Fri, 10 Dec 2010 17:50:25 GMT
View Forum Message <> Reply to Message

Hi all,

Quote:
Actually, on second thoughts I think the code as written is fine without volatile because the mutex forces the shared data to be updated in memory. Possibly an atomic variable that is read or written outside of a mutex region is more likely to need volatile.

Yes this is the case, and is what I explained in my earlier post.

Maybe what you need Koldo, in the example you gave is something like this:

```
#include <CtrlLib/CtrlLib.h>

using namespace Upp;
class AtomicVar
{
 private:
  Atomic val;
 public:
  AtomicVar() {};

  AtomicVar( AtomicVar& p) { AtomicWrite(val, p); }
  template <class T>
  AtomicVar& operator=(const T& p) { AtomicWrite(val, p); }

  operator int() {return AtomicRead(val); }

};

GUI_APP_MAIN
{

 AtomicVar v;

 v=5;

 String str = "Value = ";

 str << (int)v;

 LOG(str);
```

}

All the accesses to 'v' are atomic, so in you're case all you have to do is replace
Atomic myRunProcess = true;
with
AtomicVar myRunProcess = true;

and just use myRunProcess normaly, without worrying about Atomic considerations !!