Freescale MQX RTOS Example Guide

lwsem_usr

This document describes the lwsem_usr example which demonstrates use of memory protection and lightweight semaphore synchronization objects in the user-mode tasks (Kinetis platforms only).

Pre-requisities

This example requires the support for User Mode to be compiled in the MQX kernel. Edit the <mqx_installation>/config/<board>/user_config.h file and add the

```
#define MQX_ENABLE_USER_MODE 1
```

Then rebuild the MQX as described in the MQX Getting Started document. **Also make sure the board jumpers are set properly as described in this document.**

Note that this application uses a different linker command file than the other examples which do not make use of User mode. The linker file for User mode applications defines additional memory areas for different levels of protection as well as the area for the memory heap used by User mode tasks. The linker files suitable for the User mode examples are named with the *_usr* postfix.

Note that not all build tools and processor platforms are supported by the User-mode feature.

More Reading

Read more information about the User vs. Privileged execution mode and about memory protection in the MQX RTOS User Guide. See also User mode API in the MQX API Reference Manual. The User mode functions all have the *_usr* prefix.

The Example

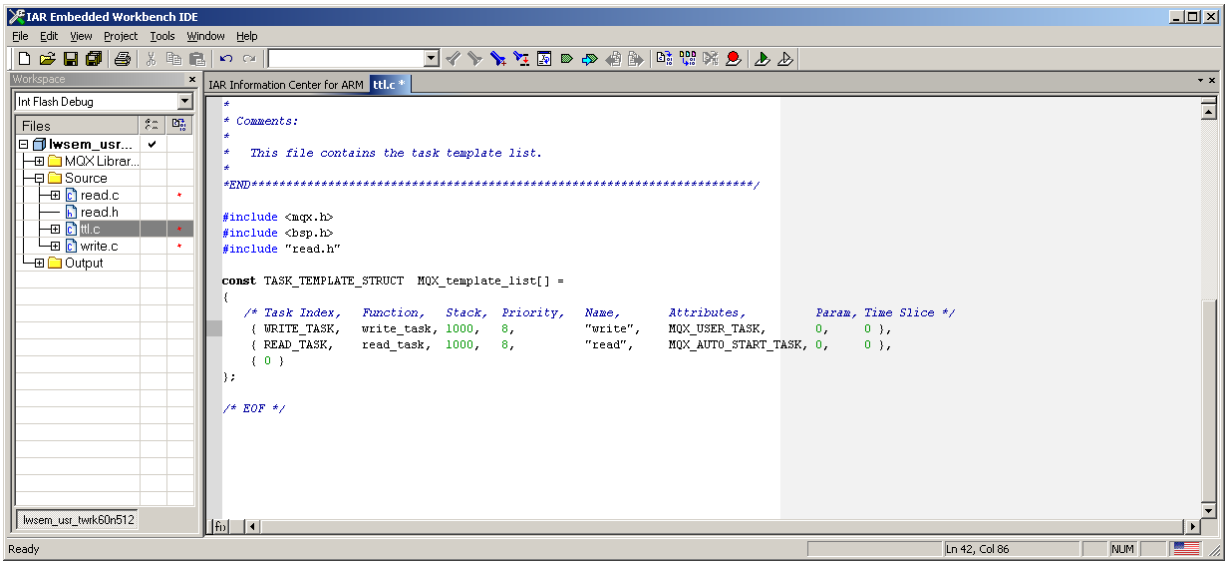
The code is based on the original "lwsem" example (which is based on the full "sem" example). It demonstrates use of lightweight semaphore objects in a multitasking application. In case of the lwsem_usr example, the "writer" tasks are running in a restricted User mode to demonstrate inter-task synchronization between User mode and Privilege modes.

The example uses the following tasks as defined in the ttl.c file:

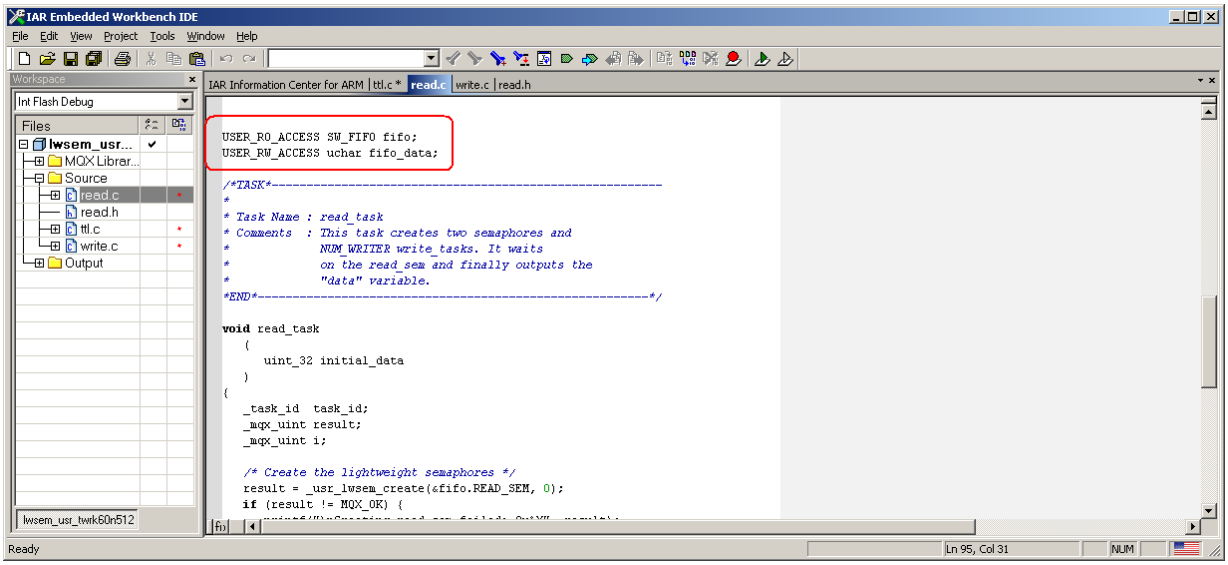
- **read_task**: The auto-start task which creates two lightweight semaphore objects. It also starts *NUM_WRITERS* number of the writer tasks. This task waits on the *READ_SEM* semaphore and prints the character passed through a global variable *fifo_data*. This variable is set as the ID of writer task which was successful in acquiring the *WRITE_SEM*.
  The read task prints the task ID and posts the *WRITE_SEM* semaphore to enable another instance of the writer task to acquire it.
- **write_task**: runs in NUM_WRITERS number of instances. It tries to acquire the WRITE_SEM, puts its ID to a global variable and posts the READ_SEM to wake up the read_task.

If everything runs correctly, the write tasks are processed in the order as they were created and the console will display the output of A B C A B C A B C...

In the ttl.c file, you can see the write_task is declared as User task with the MQX_USER_TASK flag. The read task remains to be a Privilege mode task as it prints the console output (printf accesses the serial driver so the execution from User task is not permitted).



Note that the lightweight semaphore (and any other synchronization object) must be declared as read-only for the User mode tasks. This assures the objects will not be accidentally or knowingly corrupted by any user task. Such a memory corruption could potentially destabilize the kernel.



The kernel performs explicit access checks on the memory used by the object. It denies initializing the component if the memory is not in the read-only area for User tasks.

Run the application, and see the console output. The output should be the same as in the standard “lwsem” example.