

Freescal e MQX RTOS Example Guide

Semaphore example

This document explains the semaphore example, what to expect when running it and a brief introduction to the API.

The example

The semaphore example code shows how semaphore works. The code is written in a way that three different semaphores are synchronized to ensure mutual exclusion of a common memory space.

Running the example

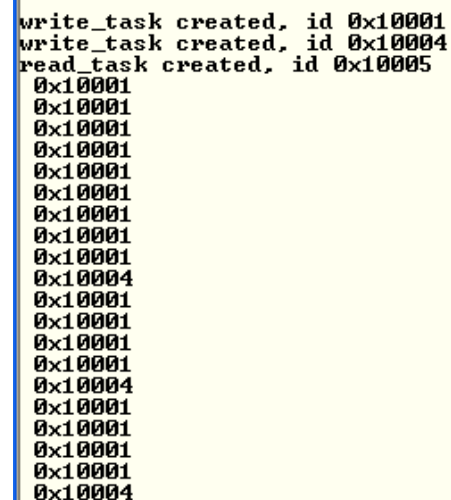
The `MQX_USE_SEMAPHORES` macro must be set to non-zero in the `user_config.h` file prior to compilation of MQX kernel libraries and the example itself.

```
#define MQX_USE_SEMAPHORES 1
```

To run the example the corresponding IDE, compiler, debugger and a terminal program are needed.

Start a terminal application on your PC and set the serial connection for 115200 baud, 8 data bits, 1 stop bit, no parity and no flow control.

After running, the results will be as the picture below.

A screenshot of a terminal window with a yellow background and a blue border. It displays the output of a program running on a Freescale MQX RTOS. The output shows the creation of three tasks: 'write_task' (ID 0x10001), 'write_task' (ID 0x10004), and 'read_task' (ID 0x10005). Following this, there is a sequence of hexadecimal values representing task IDs. The sequence starts with 15 instances of '0x10001', followed by 1 instance of '0x10004', then 15 instances of '0x10001', followed by 1 instance of '0x10004', and finally 1 instance of '0x10001'. This sequence demonstrates the execution order of the tasks, showing that the 'write_task' (ID 0x10001) and 'read_task' (ID 0x10005) execute in an interleaved manner, with the 'write_task' (ID 0x10004) executing in between.

```
write_task created, id 0x10001
write_task created, id 0x10004
read_task created, id 0x10005
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10004
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10001
0x10004
0x10001
0x10001
0x10001
0x10001
0x10001
0x10004
```

Explaining the example

The application example creates three tasks with same priority and FIFO policy.

Main task

This task initializes three semaphores (`write_sem`, `read_sem`, `index_sem`), creates `NUM_WRITERS` `write_task`'s, and creates one `read_task`.

Write task

This task opens a connection to all three semaphores then waits for sem.write and sem.index.

If the write_sem and index_sem are available, the write_task writes one entry in the data array (id of active write task) and posts sem.index and sem.read.

Read task

This task opens a connection to all three semaphores then waits for sem.read and sem.index.

If the read_sem and index_sem are available, the read_task will displays element in the data array. Sem.index and sem.write are then posted.

Flow chart

