

## Freescal MQX RTOS Example Guide

### lwsem\_lite example

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

### The example

The lwsem\_lite example code shows how lightweight semaphores works. The code is written in a way that two different semaphores are synchronized to ensure mutual exclusion of a common memory space.

### Running the example

The user only needs to do compilation of MQX libraries, ksdk library and the example without any further step.

If the platform supports floating point, you have to disable floating point in `<MQX_folder>\rtos\mqx\config\mcu\<board>\mqx_sdk_config.h`:

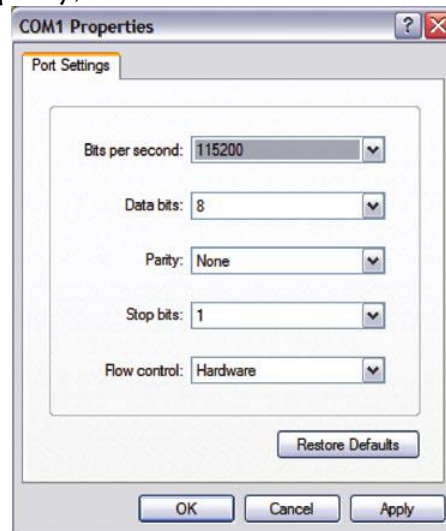
```
#define MQXCFG_ENABLE_FP      0
```

And rebuild MQX library.

Start HyperTerminal on the PC (Start menu->Programs->Accessories->Communications). Make a connection to the serial port that is connected to the board (usually will be COM1).



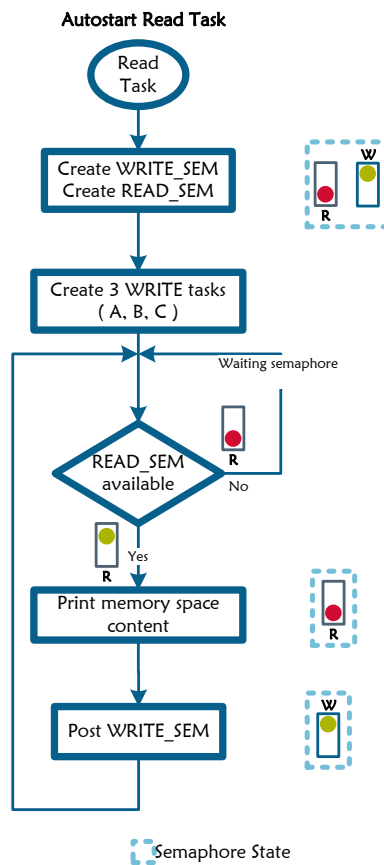
Set it for 115200 baud, no parity, 8 bits and click OK.



### Explaining the example

The light weight semaphores example uses four tasks:

- 1 read\_task
- 3 write\_task



read\_task flow chart

The read\_task starts by creating two lightweight semaphores (WRITE\_SEM and READ\_SEM) that govern the access to a data memory location (FIFO.Data). The WRITE\_SEM starts enabled, while the READ\_SEM is disabled.

```
Result = _lwsem_create(&fifo.READ_SEM, 0);
Result = _lwsem_create(&fifo.WRITE_SEM, 1);
```

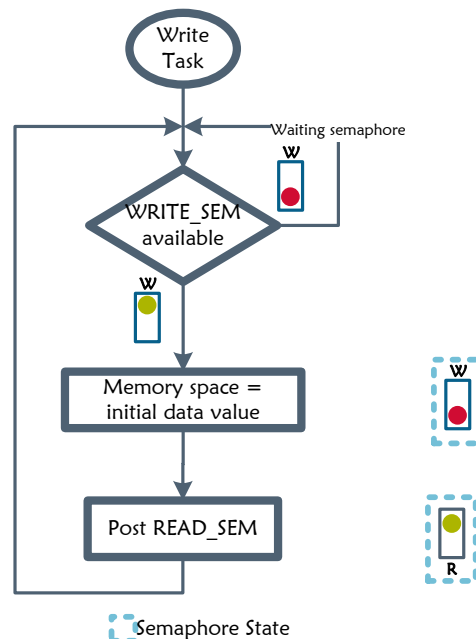
Parameters:  
 &fifo.READ\_SEM = pointer to the lightweight semaphore to create  
 0 = Initial semaphore counter

Then 3 write task are created with initial\_data equal to 'A', 'B', 'C' correspondingly.

If the READ\_SEM is available (\_lwsem\_wait), the read\_task prints on the HyperTerminal whatever the shared memory space contains.

```
Result = _lwsem_wait(&fifo.READ_SEM);
putchar('\n');
putchar(fifo.DATA);
```

Finally the WRITE\_SEM is posted  
 \_lwsem\_post(&fifo.WRITE\_SEM);



write\_task flow chart

The write task verifies if the WRITE\_SEM is available (\_lwsem\_wait), then writes at the shared memory space, task initial\_value.

```
Result = _lwsem_wait(&fifo.WRITE_SEM)
```

