Freescale MQX RTOS Example Guide

Mutex_lite example

This document explains the mutex_lite example, what to expect from the example and a brief introduction to the API.

The example

The mutex_lite example code is used to demonstrate how to use a mutex to synchronize two tasks. It creates a mutex and two tasks. Both tasks print out messages to a common terminal. Each task will lock the mutex before printing and unlock it after printing to ensure that the outputs from tasks are not mixed together on terminal.

Running the example

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

In <MQX_folder>\rtos\mqx\config\mcu\<board>\mqx_sdk_config.h please set
```
#define MQX_USE_MUTEXES                 1
#define MQXCFG_ALLOCATOR                MQX_ALLOCATOR_LWMEM
```

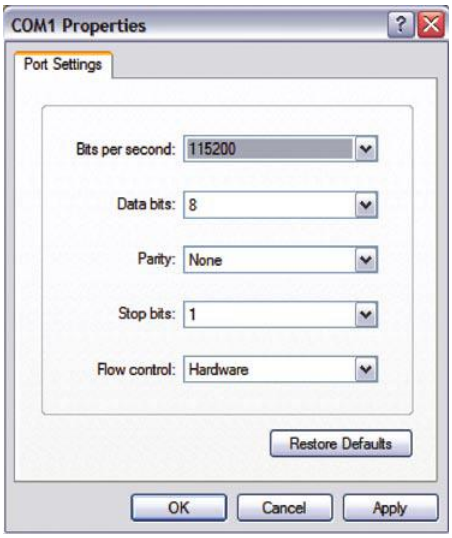
If the platform supports floating point, you have to disable floating point:

```
#define MQXCFG_ENABLE_FP                0
```

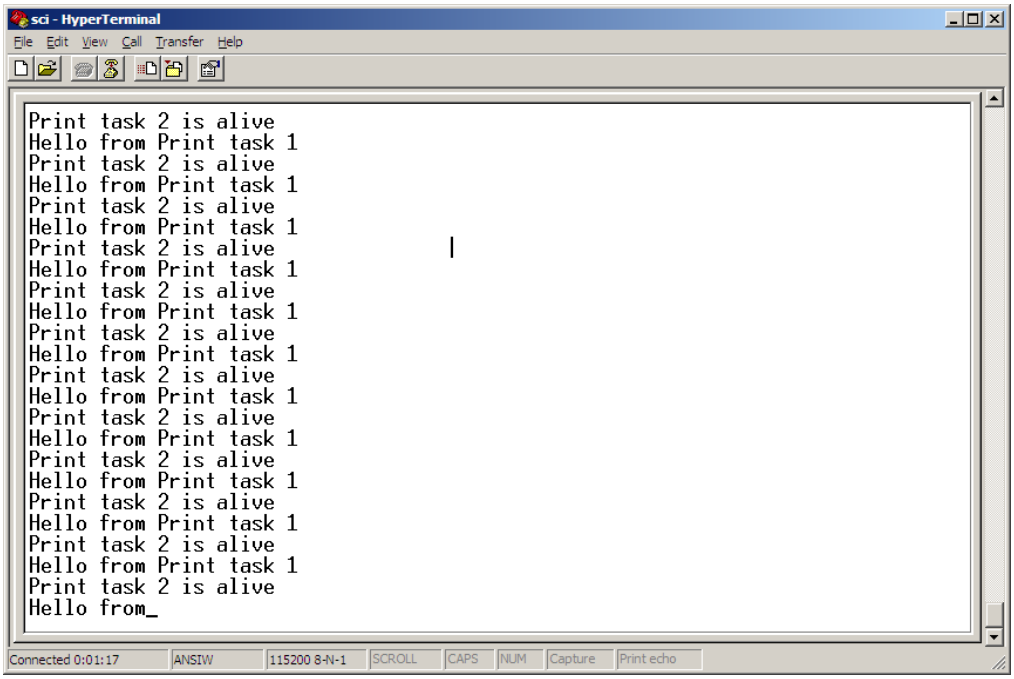
And rebuild the MQX library. This may take several minutes depending on the speed of the PC.

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.

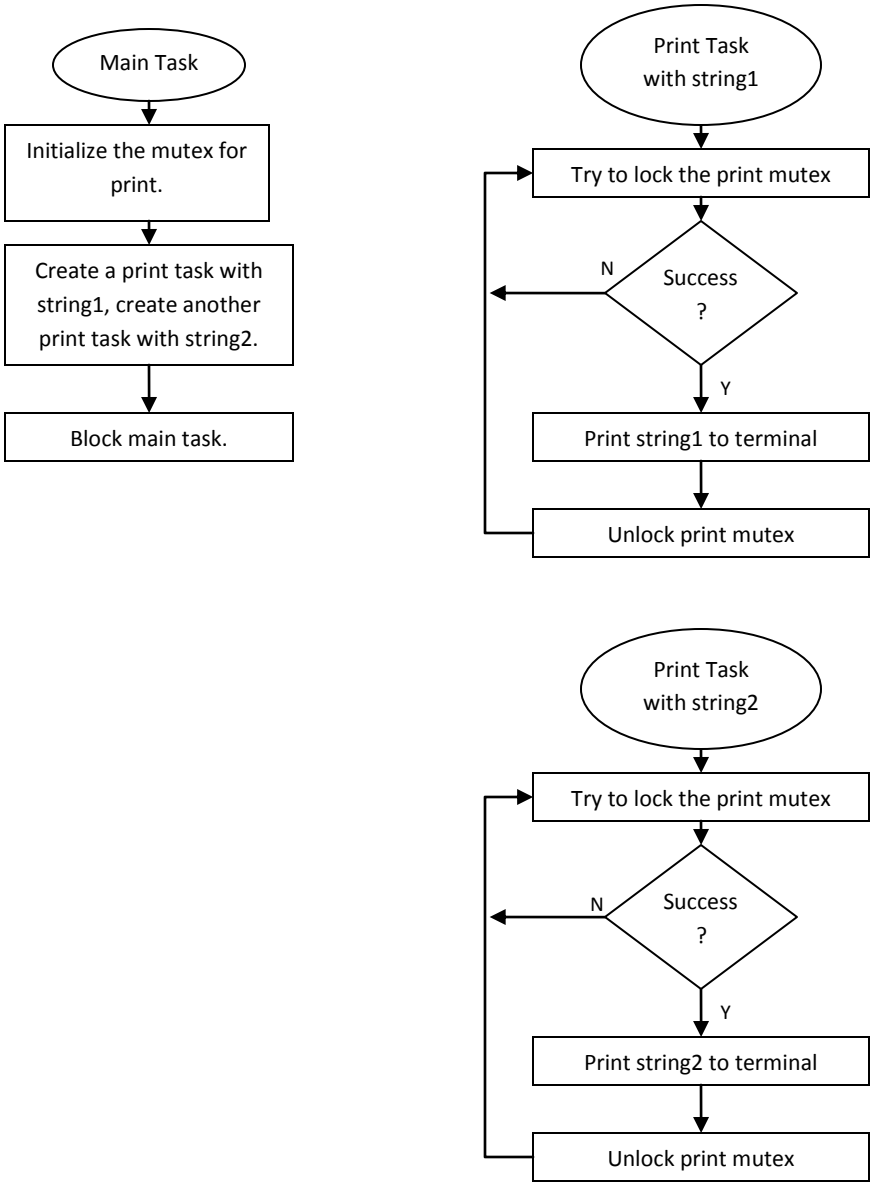


Then the mutex_lite example can be compiled and downloaded to target board. We can see the running results displayed in the HyperTerminal:



Explanation of the example

The application demo creates the main task first. The main task then creates two print tasks. The flows of the tasks are described in the next figure.



The main task first initializes the mutex with the following line:
_mutex_init(&print_mutex, &mutexattr)

Then the main task creates a print task with parameter strings1, and creates a second print task with parameter strings2.
Then the main task blocks itself.

The two print tasks both use the terminal output. If they used it in the same time, there would be conflicts, so a mutex is used to synchronize the two tasks.

Each print task will try to lock the mutex before printing the message; it will wait for the mutex as long as needed. Once the mutex is locked it prints the message and then unlocks the mutex so that the other task can lock it. This process is repeated indefinitely.