Freescale MQX Example Guide
Context switch callback example

This document describes the context switch callback example application. The example handles four different tasks.

Running the example

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.

Start context_switch_callback example on the target platform. For instructions how to do that in different IDEs and for different debuggers, see the MQX documentation (<MQX installation folder>/doc/tools).

After starting the application, you will see the printed message as the following.

```
[OK]: Stack overflowed - task id: 0x10001

Hello
World

Hello
World
...
```

Explanation of the example

There are four tasks in the example (INIT_TASK, OVERFLOWED_TASK, WORLD_TASK, HELLO_TASK). INIT_TASK starts automatically and try to create OVERFLOWED_TASK at specified location then end the task. If creation of OVERFLOWED_TASK is succeed, The OVERFLOWED_TASK initializes global variables, create the HELLO_TASK and WORLD_TASK. Finally, it calls a recursive function to make its stack overflowed. When the HELLO_TASK is actived, HELLO_TASK checks whether the OVERFLOWED_TASK is still alive or not, verify the proper operation of context switch handler, prints string "\nHello \n" and post a semaphore to unblock WORLD_TASK. After WORLD_TASK get the semaphore, WORLD_TASK prints string "World" and also verify the proper operation of context switch handler.

INIT_TASK

- Creates OVERFLOWED_TASK by _task_create_at function. If creating failed, error message is printed out to the console.
- Calls _task_block function to end the task.

OVERFLOWED_TASK

- Initialize global variables.
- Creates HELLO_TASK and WORLD_TASK by _task_create function. If creating failed, error message is printed out to the console.
- Call func function recursively to make stack overflowed and waiting for task switching.

HELLO_TASK:

- Check the value of overflowed_task_id which was updated in the callback should be equal to the task_id of OVERFLOWED_TASK then print out the result to the console.
- Check the OVERFLOWED_TASK is not alive. If not, an error message is printed out to the console.
- Check the callback has been called and the sequence of task switching is correct. If not, error messages are printed out to the console.
- Print out the string "\n Hello\n" by printf function.
- Post the semaphore to the WORLD_TASK by _lwsem_post function.

WORLD_TASK:

- Wait a semaphore posted from HELLO_TASK by _lwsem_wait function.
- Prints out the string "World \n" by printf function.
- Check the callback has been called, and the sequence of task switching is correct. If not, error messages are printed out to the console.