Freescale MQX Example Guide
Hello_lite example

This document describes the hello_lite example application. The example hello_lite handles two different tasks. Every task prints text to a console and ends.

Running the example

Then we compile the project event_lite.

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
#define MQXCFG_ALLOCATOR                MQX_ALLOCATOR_LWMEM
```

And rebuild MQX library.

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 hello_lite 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.

Hello
World

Explanation of the example

There are two tasks in the example (WORLD_TASK, HELLO_TASK). WORLD_TASK starts automatically and try to create higher priority task HELLO_TASK. If creation of HELLO_TASK succeed, HELLO_TASK only prints string "\nHello \n" and ends. After HELLO_TASK ended, WORLD_TASK prints string "World" and also ends.
WORLD_TASK:

- Creates HELLO_TASK by _task_create function. If creating failed, error message is printed out to the console.
- HELLO_TASK is created with higher priority and is activated after creating.
- When scheduler activates WORLD_TASK it prints out the string "World \n" by printf function.
- Calls _task_block function to end the task.

HELLO_TASK:

- After creating and activating this task, the string "\n Hello\n" is printed out by printf function.

- Calls _task_block function to end the task.