Freescale MQX RTOS Example Guide

Lwdemo example

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

The example

The example demonstrates the usage of core components of the MQX RTOS for synchronizing tasks including the light weight semaphore (lwsem), light weight message queue (lwmsgq), light weight event (lwevent).
The example defines 7 different tasks and context switch between tasks is examined in order for the user to understand how MQX task scheduler functions. With regards to the priority tasks are assigned into three groups with priority levels 9, 10 and 11 respectively. Also lwsem, lwmsgq and lwevent are used to block or run different tasks at specific moment in time.
For demonstration purpose of scheduling components only one task outputs simple dot character '.' to the terminal output. The user needs to use the task aware debugging (TAD) component of the IDE to examine the state of different tasks in time and the dependence of tasks on the scheduling components.

Running the example

The user only needs to do compilation of MQX libraries, ksdk library and the example without any further step.
To run the example the corresponding IDE, compiler, debugger and a terminal program are needed.

Explaining the example

The application example creates 7 tasks with the detail explanation as follow.

The MQX allocates tasks into three queues with corresponding priority levels 9, 10, 11. In each queue the ready tasks wait to be scheduled in the first in first out (FIFO) manner.

- Task LwSEMA and task LWSEMB have priority levels of 9 and 10 respectively and they wait for the lwsem. Therefore task LWSEMA is scheduled to run before task LWSEMB in case the lwsem is available.
- Task LWEVENTA and task LWEVENTB have similar priority level – level 9. Task LWEVENTA sets the event bit which allows task LWEVENTB to run.
- Task Sender has priority level 10 whereas task Responder's priority level is 9. The two tasks exchange messages and the task Responder has the priority to run over task Sender in case both of them have message in their message queues.
- Task main_task is blocked after it sends the message to task Sender as there is no other task exchanging message with it.

To see how MQX manages the task scheduling user should look at the task lists under the MQX item in the toolbar of the IDE.

The following output is expected on terminal.