

Freescale MQX RTOS Example Guide

Demo example

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

The example

The example demonstrates the usage of common components of the MQX RTOS for synchronizing tasks including the semaphore, mutex, message, event and the Round Robin scheduling component.

The example defines 11 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 semaphore, mutex, message and event 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 `MQX_HAS_TIME_SLICE`, `MQX_USE_SEMAPHORES`, `MQX_USE_LOGS` macros must be set to non-zero in the `user_config.h` file prior to compilation of MQX libraries and the example itself.

To run the example the corresponding IDE, compiler, debugger and a terminal program are needed.

Explaining the example

The application example creates 11 tasks with the flow control and explanation as shown in page 3 and 4.

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 MutexA and task MutexB have the same priority level - level 9 - and wait for the mutex. Hence the task is ready next is the task that has the mutex and has been waiting the longest in the task ready queue.
- Task SemA and task SemB have priority levels of 9 and 10 respectively and they wait for the semaphore. Therefore task SemA is scheduled to run before task SemB in case the semaphore is available.
- Task EventA and task EventB have similar priority level - level 9. Task EventA sets the event bit which allows task EventB 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 ATimeSliceTask and task BTimeSliceTask have the same priority level - level 9. Task ATimeSliceTask is scheduled to run only one time slot of 50 ms because the Round Robin component is applied to it.
- 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 up the items under the MQX in the menu bar of the IDE.

The following output is expected on terminal.

```
MQX 4.1.0  
Hello from main_task(<).  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....  
.....
```



