Freescale MQX RTOS Example Guide
lwdemo_usr

This document describes the lwdemo_usr example which demonstrates use of memory protection and various synchronization objects in the user-mode tasks (Kinetis platforms only).

Pre-requisities

This example requires the support for User Mode to be compiled in the MQX kernel. Edit the <mqx_installation>/config/<board>/user_config.h file and add the

```
#define MQX_ENABLE_USER_MODE 1
```

Then rebuild the MQX as described in the MQX Getting Started document. **Also make sure the board jumpers are set properly as described in this document.**

Note that this application uses a different linker command file than the other examples which do not make use of User mode. The linker file for User mode applications defines additional memory areas for different levels of protection as well as the area for the memory heap used by User mode tasks. The linker files suitable for the User mode examples are named with the *_usr* postfix.

Note that not all build tools and processor platforms are supported by the User-mode feature.

More Reading

Read more information about the User vs. Privileged execution mode and about memory protection in the MQX RTOS User Guide. See also User mode API in the MQX API Reference Manual. The User mode functions all have the *_usr* prefix.

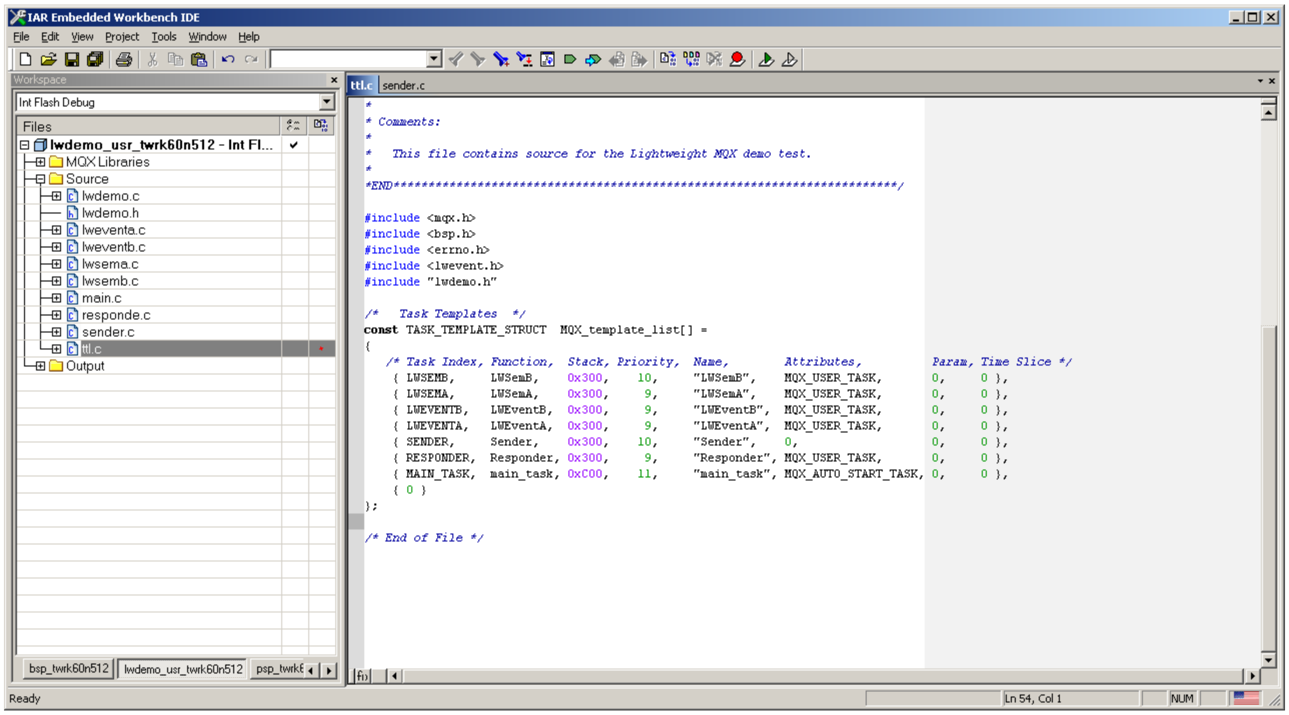
The Example

The code is based on the original “lwdemo” example (which is based on the full “demo” example). It demonstrates use of various lightweight synchronization and message objects in a multi-tasking application. In case of the lwdemo_usr example, some of the tasks are running in a restricted User mode to demonstrate inter-task synchronization and communication between User mode and Privilege modes.

The example uses the following tasks as defined in the ttl.c file:

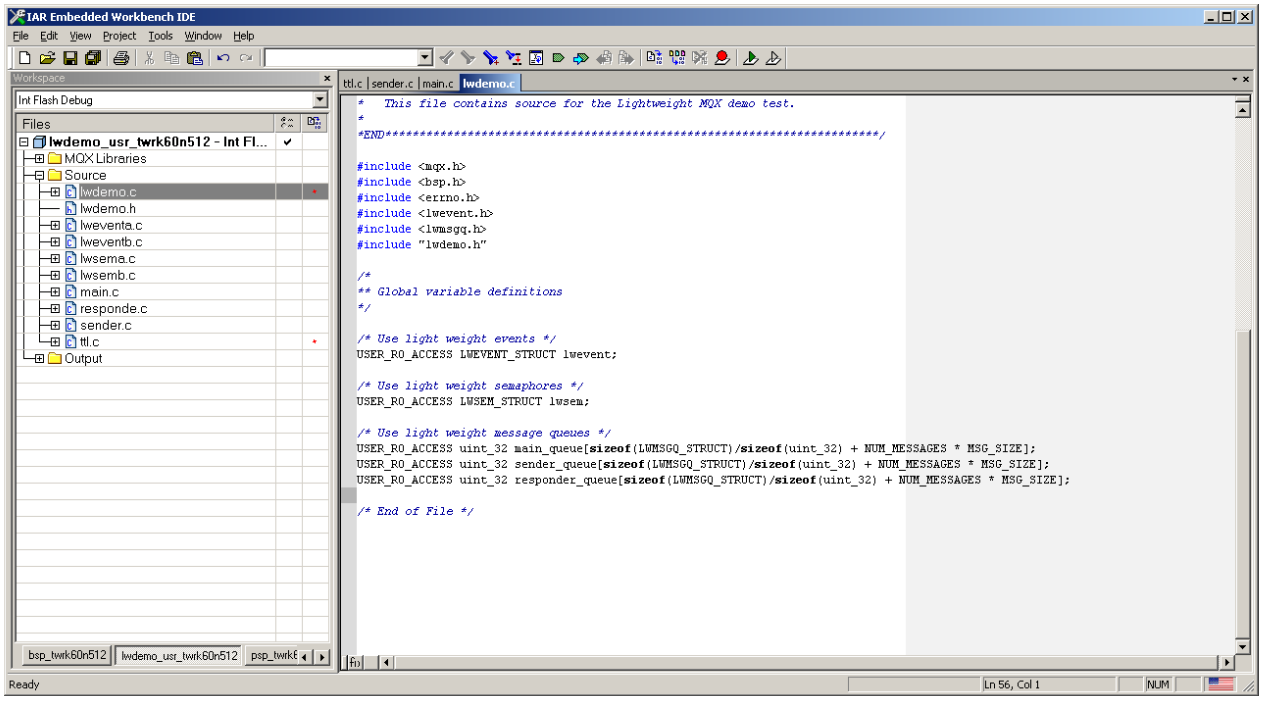
- **main_task**: The auto-start task which creates all necessary synchronization objects. This task also sends the initial message to the *sender_queue* which triggers the Sender-Responder message exchanging. Finally the main_task remains blocked in a message receive call as no other task ever sends a message to the main_task’s *main_queue*.
- **Sender** task: forwards the message received from *sender_queue* to the *responder_queue*.
- **Responder** task: forwards the message received from *responder_queue* to the *sender_queue*. It prints a dot any time the message is handled. The Sender and Responder tasks are cyclically exchanging the single message object.
- **LWSemA** task: acquires the *lwsem* semaphore object and posts it again.
- **LWSemB** task: acquires the same *lwsem* semaphore object and posts it again. The LWSemA and LWSemB tasks effectively demonstrate mutual exclusion by using a lightweight semaphore object.
- **LWEventA** task: periodically sets the *lwevent* lightweight event object.
- **LWEventB** task: waits **and** clears the *lwevent* object. The LWEventA and LWEventB tasks demonstrate task notification by using a lightweight event object.

In the ttl.c file, you can see what the application tasks are declared as User tasks with the MQX_USER_TASK flag. The Sender task remains to be a Privilege mode task as it prints the console output (printf accesses the serial driver so the execution from User task is not permitted).



```
*
* Comments:
*
*   This file contains source for the Lightweight MQX demo test.
*
*END************************************************************************/

#include <mqx.h>
#include <bsp.h>
#include <errno.h>
#include <lwevent.h>
#include "lwdemo.h"

/*   Task Templates  */
const TASK_TEMPLATE_STRUCT  MQX_template_list[] =
{
   /* Task Index, Function,  Stack, Priority,  Name,        Attributes,          Param, Time Slice */
    { LWSEMB,     LWSemB,    0x300,    10,     "LWSemB",    MQX_USER_TASK,       0,     0 },
    { LWSEMA,     LWSemA,    0x300,     9,     "LWSemA",    MQX_USER_TASK,       0,     0 },
    { LWEVENTB,   LWEventB,  0x300,     9,     "LWEventB",  MQX_USER_TASK,       0,     0 },
    { LWEVENTA,   LWEventA,  0x300,     9,     "LWEventA",  MQX_USER_TASK,       0,     0 },
    { SENDER,     Sender,    0x300,    10,     "Sender",    0,                   0,     0 },
    { RESPONDER,  Responder, 0x300,     9,     "Responder", MQX_USER_TASK,       0,     0 },
    { MAIN_TASK,  main_task, 0xC00,    11,     "main_task", MQX_AUTO_START_TASK, 0,     0 },
    { 0 }
};

/* End of File */
```

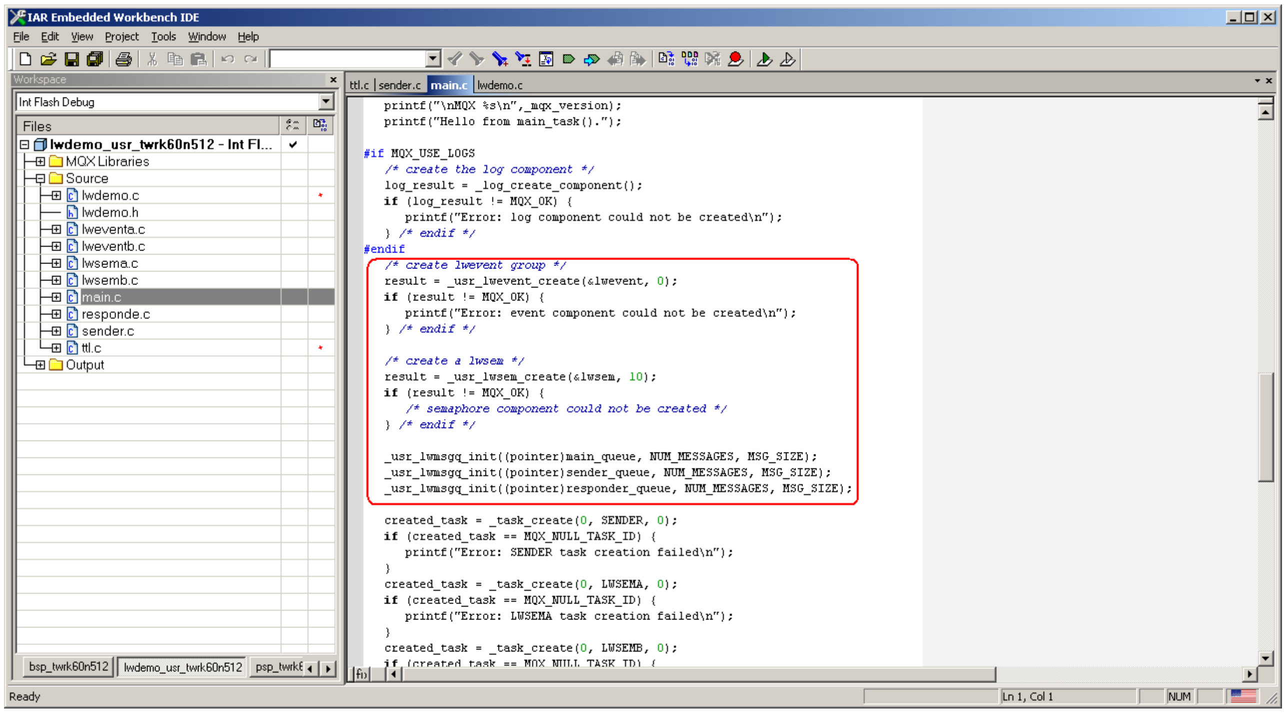
Note that the synchronization objects and message queues must be declared as read-only for the User mode tasks. This assures the objects will not be accidentally or knowingly corrupted by any user task. Such a memory corruption could potentially destabilize the kernel.



```
*   This file contains source for the Lightweight MQX demo test.
*
*END************************************************************************/

#include <mqx.h>
#include <bsp.h>
#include <errno.h>
#include <lwevent.h>
#include <lwmsgq.h>
#include "lwdemo.h"

/*
** Global variable definitions
*/

/* Use light weight events */
USER_RO_ACCESS LWEVENT_STRUCT lwevent;

/* Use light weight semaphores */
USER_RO_ACCESS LWSEM_STRUCT lwsem;

/* Use light weight message queues */
USER_RO_ACCESS uint_32 main_queue[sizeof(LWMSGQ_STRUCT)/sizeof(uint_32) + NUM_MESSAGES * MSG_SIZE];
USER_RO_ACCESS uint_32 sender_queue[sizeof(LWMSGQ_STRUCT)/sizeof(uint_32) + NUM_MESSAGES * MSG_SIZE];
USER_RO_ACCESS uint_32 responder_queue[sizeof(LWMSGQ_STRUCT)/sizeof(uint_32) + NUM_MESSAGES * MSG_SIZE];

/* End of File */
```

The kernel performs explicit access checks on the memory used by the object. It denies initializing the component if the memory is not in the read-only area for User tasks.

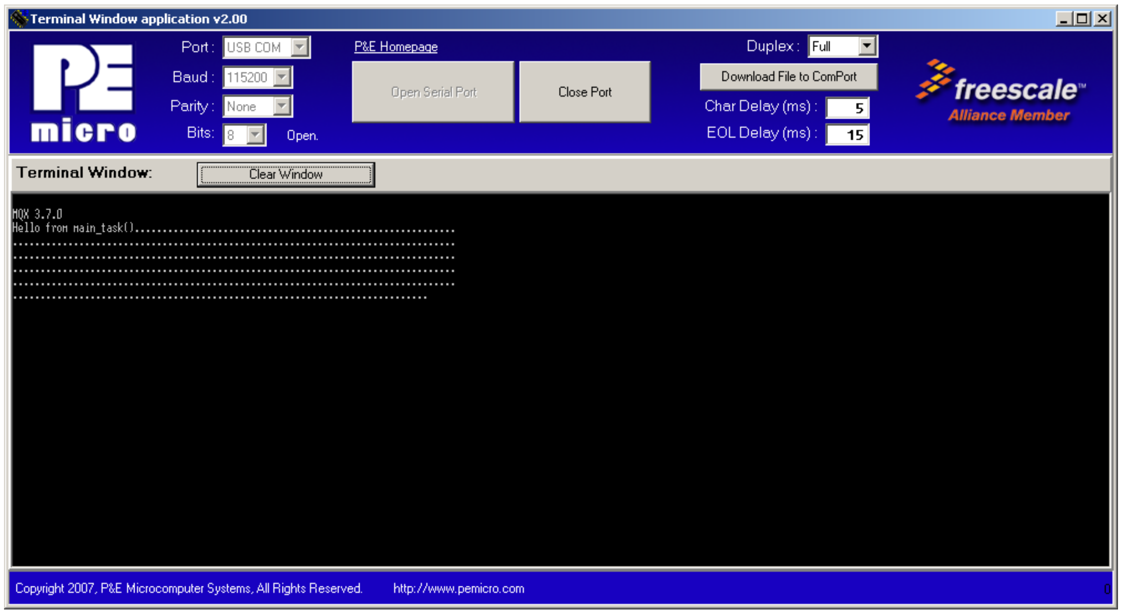
Open the main.c file:



```
   printf("\nMQX %s\n",_mqx_version);
   printf("Hello from main_task().");

#if MQX_USE_LOGS
   /* create the log component */
   log_result = _log_create_component();
   if (log_result != MQX_OK) {
      printf("Error: log component could not be created\n");
   } /* endif */
#endif
   /* create lwevent group */
   result = _usr_lwevent_create(&lwevent, 0);
   if (result != MQX_OK) {
      printf("Error: event component could not be created\n");
   } /* endif */

   /* create a lwsem */
   result = _usr_lwsem_create(&lwsem, 10);
   if (result != MQX_OK) {
      /* semaphore component could not be created */
   } /* endif */

   _usr_lwmsgq_init((pointer)main_queue, NUM_MESSAGES, MSG_SIZE);
   _usr_lwmsgq_init((pointer)sender_queue, NUM_MESSAGES, MSG_SIZE);
   _usr_lwmsgq_init((pointer)responder_queue, NUM_MESSAGES, MSG_SIZE);

   created_task = _task_create(0, SENDER, 0);
   if (created_task == MQX_NULL_TASK_ID) {
      printf("Error: SENDER task creation failed\n");
   }
   created_task = _task_create(0, LWSEMA, 0);
   if (created_task == MQX_NULL_TASK_ID) {
      printf("Error: LWSEMA task creation failed\n");
   }
   created_task = _task_create(0, LWSEMB, 0);
   if (created_task == MQX_NULL_TASK_ID) {
```

Note that the synchronization objects are initialized with the *_usr_* API calls which is mandatory when the object is to be accessed by a User mode task.

Run the application, and see the console output. The output should be the same as in the "lwdemo" example. The Sender (privileged) task prints dots '.' periodically when it receives and exchanges a message with the Responder task (which runs in the User-mode).


Terminal Window application v2.00
P&E Homepage
Open Serial Port
Close Port
Download File to ComPort
Terminal Window:
Clear Window
freescale Alliance Member
Copyright 2007, P&E Microcomputer Systems, All Rights Reserved.
http://www.pemicro.com

Port : USB COM
Baud : 115200
Parity : None
Bits: 8
Open.
Duplex : Full
Char Delay (ms) : 5
EOL Delay (ms) : 15

```
MQX 3.7.0
Hello from main_task()..........................................................
................................................................................
................................................................................
................................................................................
................................................................................
..........................................................................
```