Assignment 3 Feedback

Winter 2014

Common Problems

- No discussion of relevance to lecture content
  - Minus one point (e.g. A instead of A+)
- Discussion of task management could have been more detailed
  - To level of task API (not necessarily beyond)
  - What is done to set up context switching?
- Discussion at semaphore/mutex API level
  - Not much about the implementation
- Misleading statements or confusing wording
  - Incorrect descriptions?
- Just not enough info

```
// TAKE
signed portBASE_TYPE xQueueGenericReceive( …   )
{
     for(;;)
    {
       taskENTER_CRITICAL();
       {
              if( pxQueue->uxMessagesWaiting > ( unsigned portBASE_TYPE ) 0 )
              {     //  message in queue!! Can take semaphore
                  prvCopyDataFromQueue( pxQueue, pvBuffer );   // read message ← really needed?

                  // decrement message count to indicate semaphore taken
                  --( pxQueue->uxMessagesWaiting );

                  #if ( configUSE_MUTEXES == 1 )
                  {
                  if( pxQueue->uxQueueType == queueQUEUE_IS_MUTEX )
                  {    // record current holder
                       pxQueue->pxMutexHolder = xTaskGetCurrentTaskHandle();
                  }
                  }
                  #endif
                  // Check if a blocked sender is waiting for the emptied space
                  if( listLIST_IS_EMPTY( &( pxQueue->xTasksWaitingToSend ) ) == pdFALSE )
                  // will this ever happen for a binary semaphore or mutex???
                  {
                     // snip!
                  }
                  //  all done … got semaphore!
                  taskEXIT_CRITICAL();
                  return pdPASS;
              }
              else   //cannot take semaphore
```

```
        else   // cannot take semaphore
        {
          if( xTicksToWait == ( portTickType ) 0 )
          {     // 0 block time specified
                taskEXIT_CRITICAL();
                return errQUEUE_EMPTY;  // FAIL!
          }
          else if( xEntryTimeSet == pdFALSE )
          {
                vTaskSetTimeOutState( &xTimeOut );
                xEntryTimeSet = pdTRUE;
          }
          }
}
taskEXIT_CRITICAL();

// Interrupts and other tasks could now be active!!
```

```
// Interrupts and other tasks could now be active!!
vTaskSuspendAll();   // need to do some scheduling to block this caller

prvLockQueue( pxQueue );

if( xTaskCheckForTimeOut( &xTimeOut, &xTicksToWait ) == pdFALSE )
{  // oops … sloppy indent!
// anything to receive  ?
if( prvIsQueueEmpty( pxQueue ) != pdFALSE )
{   // nothing to receive
      #if ( configUSE_MUTEXES == 1 )
      {
      if( pxQueue->uxQueueType == queueQUEUE_IS_MUTEX )
      {
            portENTER_CRITICAL();
            {     // holder inherits priority!
                  vTaskPriorityInherit( ( void * ) pxQueue->pxMutexHolder );
            }
            portEXIT_CRITICAL();
      }
      }
      #endif
// NOW … block the task!
```

```
// NOW … Block the task!
            vTaskPlaceOnEventList( &( pxQueue->xTasksWaitingToReceive ), xTicksToWait );
            prvUnlockQueue( pxQueue );
            if( xTaskResumeAll() == pdFALSE )
            {
               portYIELD_WITHIN_API();
               // will eventually resume here! … either timed out, or a message in queue
             }
      }
      else
      {  // surprise … there is a message in the queue! … get on next loop (maybe)
            prvUnlockQueue( pxQueue );
            ( void ) xTaskResumeAll();
      }
      } // close sloppy indent!
      else   // timer has expired
      {
               prvUnlockQueue( pxQueue );
               ( void ) xTaskResumeAll();
               return errQUEUE_EMPTY;
      }
   }  // loop as needed!
}
```