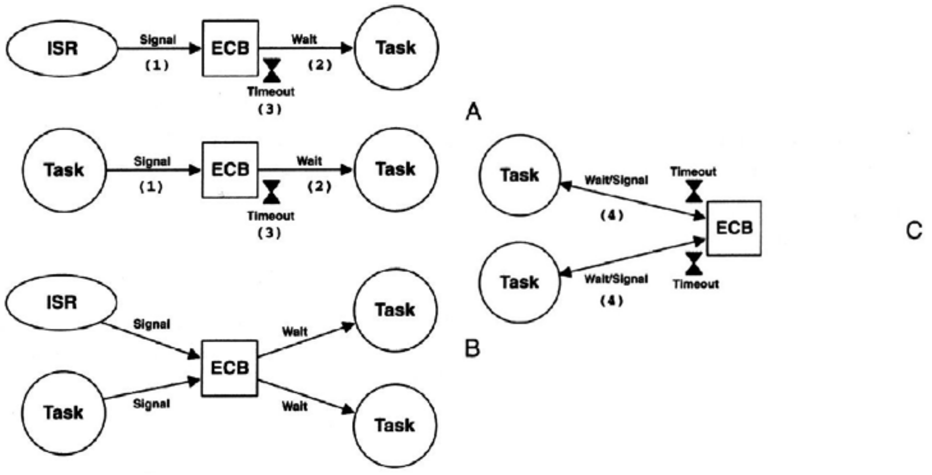
Chapter 2, Section 2.5: Architecture and Characteristics of a RTOS kernel (μC/OS-II)

1. Introduction
2. Interrupt Management
3. Task Management
4. Time Management
5. *Event Management*
6. Memory Management
7. Porting μCOS-II

5. Event Management

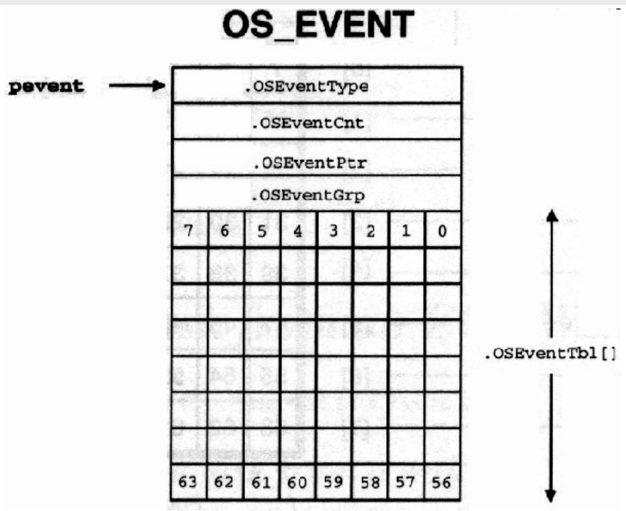
5.1 Definition of Event Control Block (ECB)
5.2 ECB Management
5.3 TCB vs ECB
5.4 Semaphore Management
5.5 Mutex Management
5.6 Mailbox Management
5.7 Message Queues Management

5.1 Definition of ECB



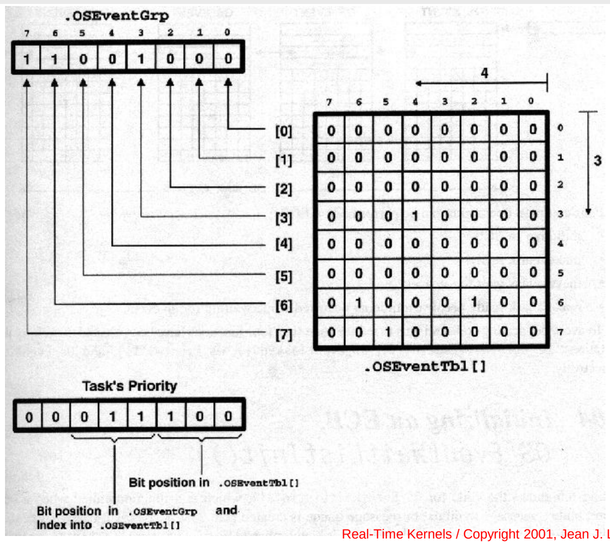
Chap 2, Section 2.5, Page 76

5.1 Definition of ECB



Chap 2, Section 2.5, Page 77

5.1 Definition of ECB


.OSEventGrp
.OSEventTbl[]
Task's Priority
Bit position in .OSEventTbl[]
Bit position in .OSEventGrp and Index into .OSEventTbl[]
[0]
[1]
[2]
[3]
[4]
[5]
[6]
[7]
4
3

Real-Time Kernels / Copyright 2001, Jean J. Labrosse

5.2 ECB Management

When *PEND* occurs, *OSEventTaskWait* will:

1. **Remove the task from OSRdyTbl and OSRdyGr**
2. **Store the task in OSEventTbl and OSEventGrp**

5.2 ECB Management

When *POST* occurs, *OSEventTaskReady* will:

1. **Elect the next task ready for the critical section (see next slide)**
2. **Remove the task from OSEventTbl and OSEventGrp**
3. **Reactivate the task in OSRdyTbl and OSRdyGrp**

Chap 2, Section 2.5, Page 80

5.2 ECB Management

When *PEND after timeout* occurs, OSEventTO:

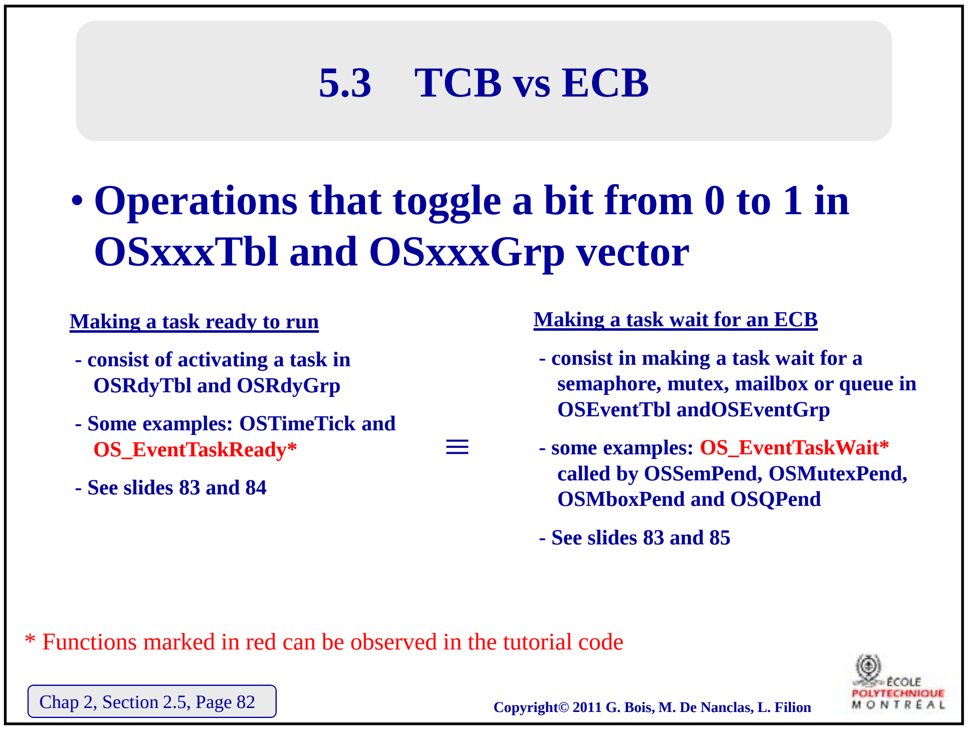
1. **Remove the task from OSEventTbl and OSEventGrp lists**

N.B. The task does not need to be reactivated into OSRdyTbl and OSRdyGrp. OSTimeTick will update OSTCBDly and the task will be reactivated after countdown reaches 0.

Chap 2, Section 2.5, Page 81

Diapositive 8

LF1 Landry Frechette; 2008-04-12



5.3 TCB vs ECB

- **Operations that toggle a bit from 0 to 1 in OSxxxTbl and OSxxxGrp vector**

Making a task ready to run

- consist of activating a task in OSRdyTbl and OSRdyGrp
- Some examples: OSTimeTick and OS_EventTaskReady*
- See slides 83 and 84

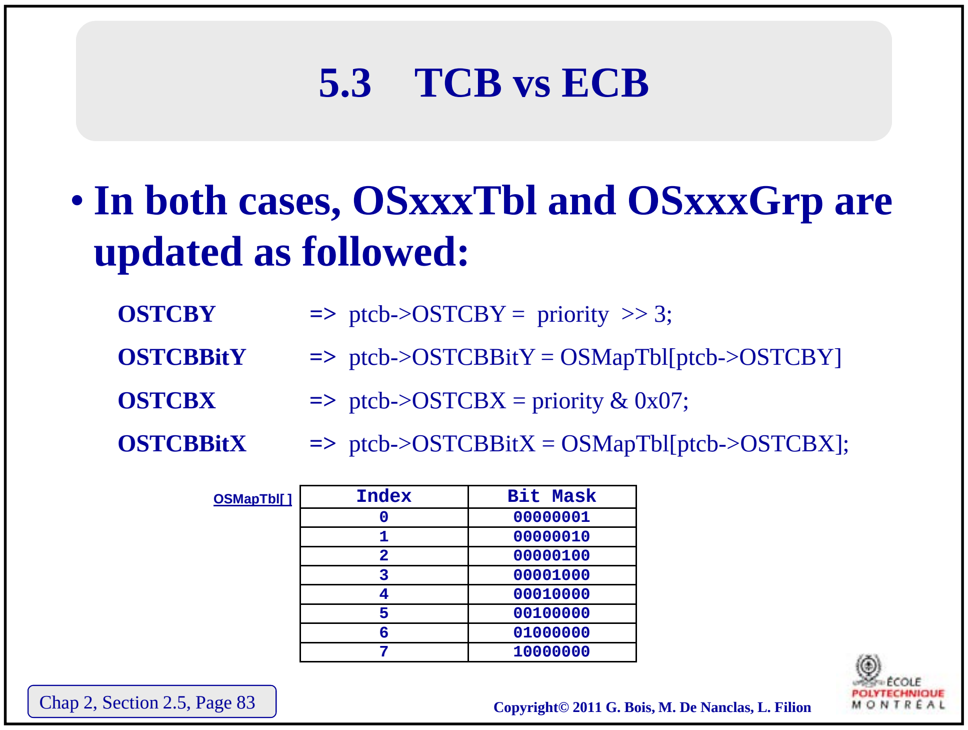
$\equiv$

Making a task wait for an ECB

- consist in making a task wait for a semaphore, mutex, mailbox or queue in OSEventTbl andOSEventGrp
- some examples: OS_EventTaskWait* called by OSSemPend, OSMutexPend, OSMboxPend and OSQPend
- See slides 83 and 85

* Functions marked in red can be observed in the tutorial code

Chap 2, Section 2.5, Page 82



5.3 TCB vs ECB

- **In both cases, OSxxxTbl and OSxxxGrp are updated as followed:**

OSTCBY	=> ptcb->OSTCBY = priority >> 3;
OSTCBBitY	=> ptcb->OSTCBBitY = OSMapTbl[ptcb->OSTCBY]
OSTCBX	=> ptcb->OSTCBX = priority & 0x07;
OSTCBBitX	=> ptcb->OSTCBBitX = OSMapTbl[ptcb->OSTCBX];

OSMapTbl[ ]

Index	Bit Mask
0	00000001
1	00000010
2	00000100
3	00001000
4	00010000
5	00100000
6	01000000
7	10000000

Chap 2, Section 2.5, Page 83

5.3 TCB vs ECB

Code to change the state of a task from the waiting state to ready to execute state:

```
OSRdyGrp                    |=      ptcb->OSTCBBitY;
OSRdyTbl[ptcb->OSTCBY]      |=      ptcb->OSTCBBitX;
```

5.3 TCB vs ECB

- **Making a task wait for an ECB:**

```
pevent->OSEventGrp                  |=      ptcb->OSTCBBitY;
pevent->OSEventTbl[ptcb->OSTCBY]    |=      ptcb->OSTCBBitX;
```

5.3 TCB vs ECB

- Operations that toggle a bit from 1 to 0 in OSxxxTbl and OSxxxGrp vector

Removing a task from the ready list

- Some examples: OSTimeDly, OSTaskSuspend, OSChangePrio, and OS_EventTaskWait
- See slides 83 and 87

≡

Removing a task from a waiting list

- OS_EventTaskReady called by SSemPost, OSMutexPost, OSMboxPost, OSQPost
- OS_EventTO called by OSSemPend, OSMutexPend, OSMboxPend, OSQPend
- See slides 83 and 88

Chap 2, Section 2.5, Page 86

5.3 TCB vs ECB

- Removing a task from the ready list :

```
if ((OSRdyTbl[ptcb->OSTCBY]  &=  ~ ptcb->OSTCBBitX) ==0)
   {OSRdyGrp  &= ~ ptcb->OSTCBBitY;
   }
```

Chap 2, Section 2.5, Page 87

5.3 TCB vs ECB

- **Removing a task from a waiting list**

```
if ((pevent->OSEventTbl[ptcb->OSTCBY]&= ~ ptcb->OSTCBBitX) ==0)
  {pevent->OSEventGrp  &= ~ ptcb->OSTCBBitY;
  }
```

Chap 2, Section 2.5, Page 88

5.3 TCB vs ECB

- **Two operations to determine the top priority task from OSxxxTbl and OSxxxGrp vector:**

Finding the highest priority task ready to run		**Finding the highest priority task waiting for the event**
- For example: e.g. OSSched, OSIntExit	$\equiv$	- For example: OS_EventTaskReady
- See slide 90		- See slide 91

Chap 2, Section 2.5, Page 89

5.3 TCB vs ECB

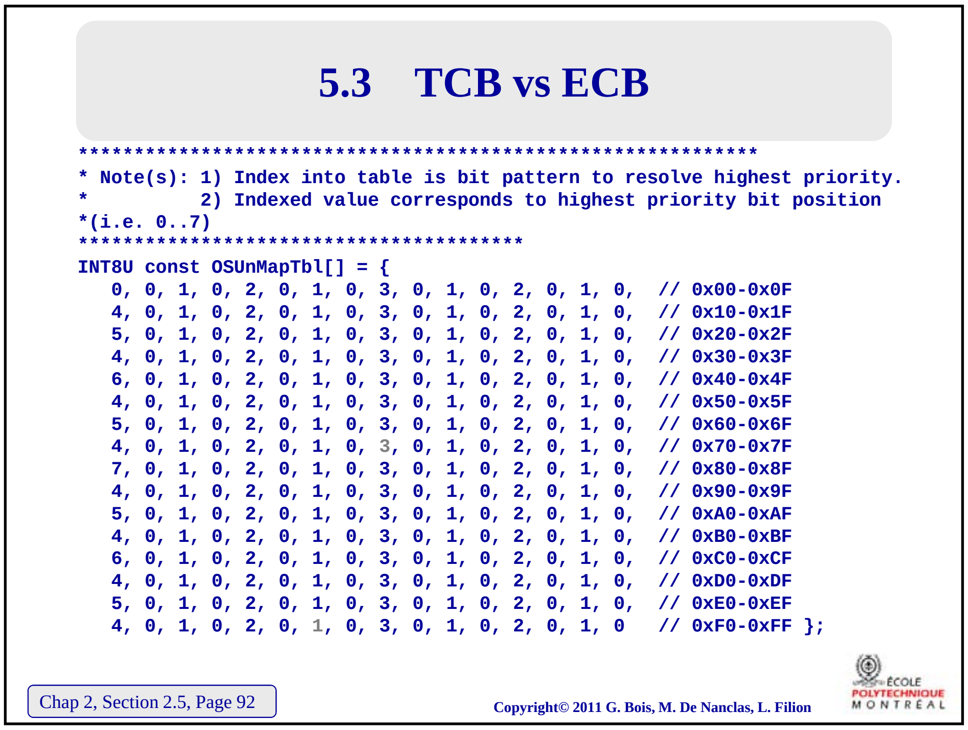
- Code to determine the top priority task (from the ready list):

```
Y       =       OSUnMapTbl[OSRdyGrp] ;
X       =       OSUnMapTbl[OSRdyTbl[Y]] ;
prio    =       [Y << 3] + X ;
```

5.3 TCB vs ECB

- Code to determine the top priority task waiting for an event:

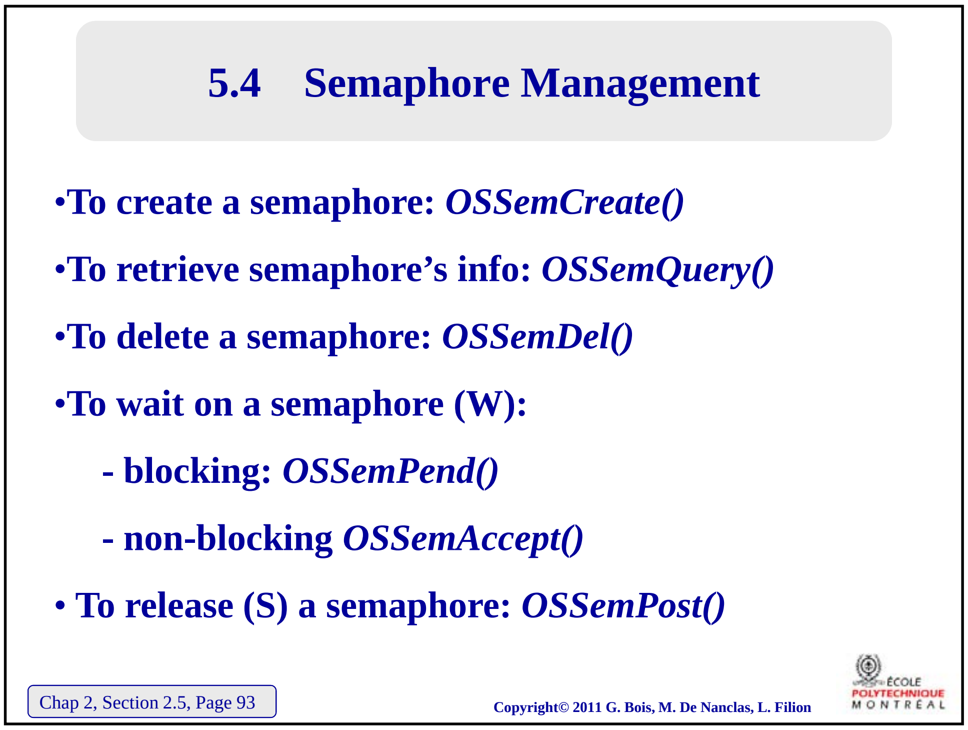
```
Y       =       OSUnMapTbl[pevent->OSEventGrp] ;
X       =       OSUnMapTbl[pevent -> OSEventTbl [Y]] ;
prio    =       [Y << 3] + X ;
```



5.3 TCB vs ECB

```
*****************************************************************
* Note(s): 1) Index into table is bit pattern to resolve highest priority.
*          2) Indexed value corresponds to highest priority bit position
*(i.e. 0..7)
******************************************
INT8U const OSUnMapTbl[] = {
   0, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x00-0x0F
   4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x10-0x1F
   5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x20-0x2F
   4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x30-0x3F
   6, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x40-0x4F
   4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x50-0x5F
   5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x60-0x6F
   4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x70-0x7F
   7, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x80-0x8F
   4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0x90-0x9F
   5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0xA0-0xAF
   4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0xB0-0xBF
   6, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0xC0-0xCF
   4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0xD0-0xDF
   5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0,  // 0xE0-0xEF
   4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0   // 0xF0-0xFF };
```

Chap 2, Section 2.5, Page 92



5.4 Semaphore Management

•To create a semaphore: *OSSemCreate()*

•To retrieve semaphore's info: *OSSemQuery()*

•To delete a semaphore: *OSSemDel()*

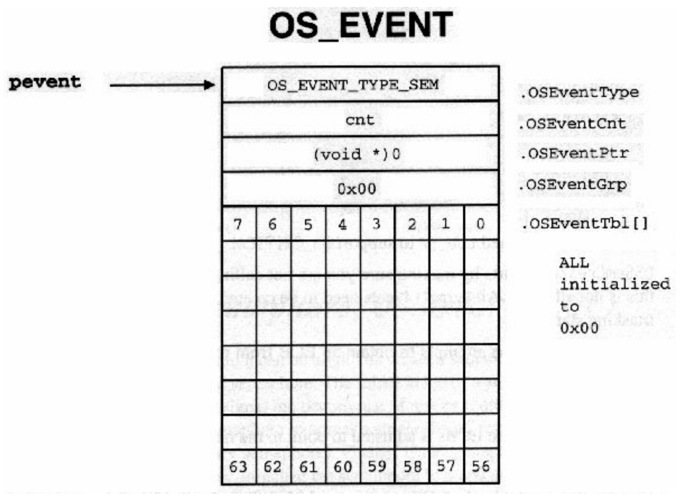
•To wait on a semaphore (W):

- blocking: *OSSemPend()*
- non-blocking *OSSemAccept()*

• To release (S) a semaphore: *OSSemPost()*

Chap 2, Section 2.5, Page 93

5.4 Semaphore Management



Real-Time Kernels / Copyright 2001, Jean J. Labrosse

5.5 Mutex Management

•To create a Mutex: *OSMutexCreate()*

•To retrieve mutexes' info: *OSMutexQuery()*

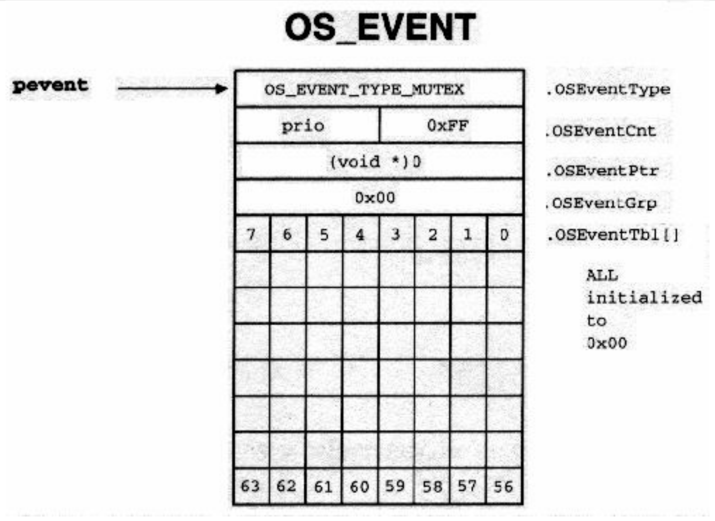
•To delete a Mutex *OSMutexDel()*

•To wait (W) on a mutex:

- blocking: *OSMutexPend()*
- non-blocking: *OSMutexAccept()*

•To release (S) a mutex: *OSMutexPost()*

5.5 Mutex Management

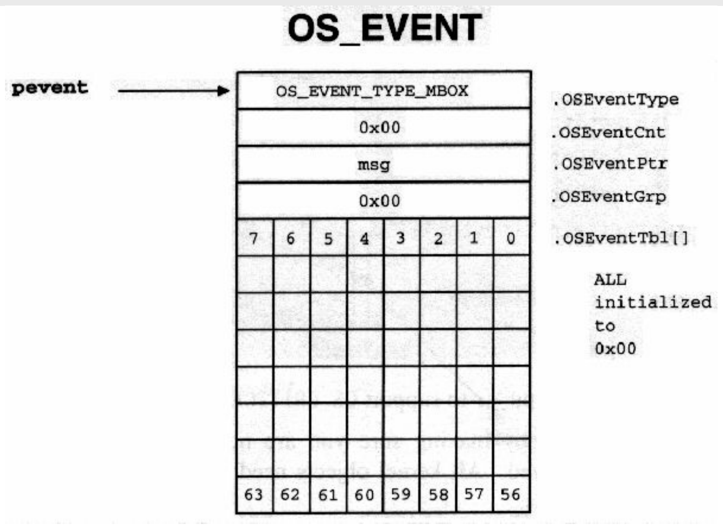


5.6 Mailbox Management

- •To create a mailbox: *OSMboxCreate()*
- •To retrieve malibox' info: *OSMboxQuery()*

  (e.g. is there data in the mailbox?)
- •To delete a mailbox: *OSMboxDel()*

5.6 Mailbox Management



Real-Time Kernels / Copyright 2001, Jean J. Labrosse

5.6 Mailbox Management

•To send a non-blocking message...

- in a single mailbox: OSMboxPost()

- to all waiting tasks: *OSMboxPostOpt()*

(we can send to the top priority waiting task, or to all tasks, e.g. broadcast)

•To read from a mailbox (take a message):

- blocking: *OSMboxPend()*

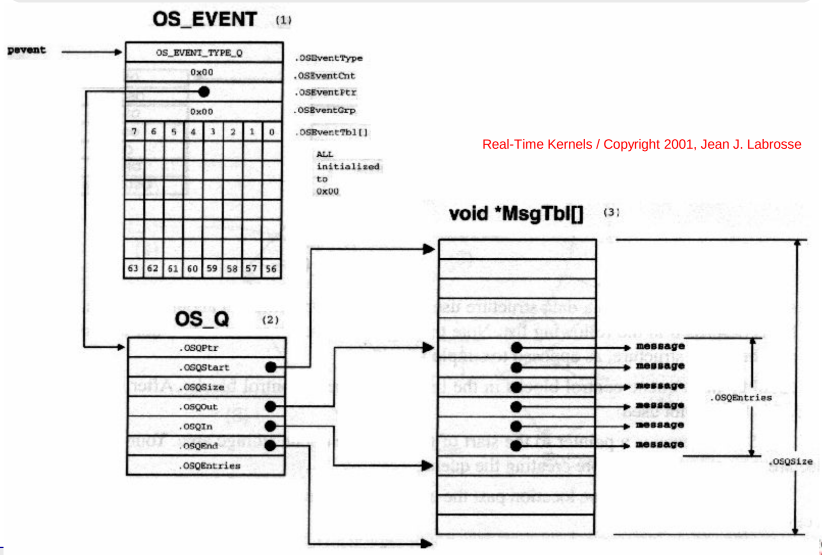
- non-blocking: *OSMboxAccept()*

5.7 Message Queues Management

- To create a message queue: ***OSQCreate()***
- To retrieve queue's info: ***OSQQuery()***
- To delete a queue: ***OSQDel()***
- To clear a queue: ***OSQFlush()***

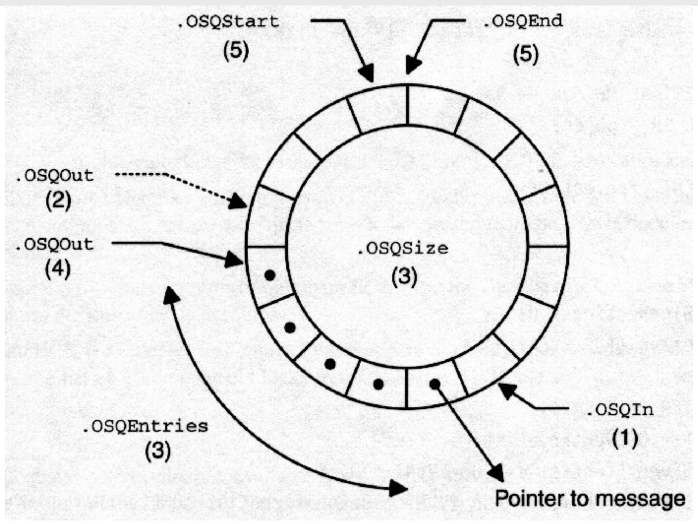
Chap 2, Section 2.5, Page 100

5.7 Message Queues Management



Chap 2, Section 2.5, Page 101

5.7 Message Queues Management



5.7 Message Queues Management

- To send a non-blocking message...
  - in a FIFO queue: *OSQPost()*
  - in a LIFO queue: *OSQPostFront()*
  - to all waiting tasks: *OSQPostOpt()*
- To read from a queue:
  - blocking: *OSQPend()*
  - non-blocking: *OSQAccept()*