Chapter 10
Message Mailbox Management

Ming-Hsin Ho

2008/8/12

Outline (1/2)

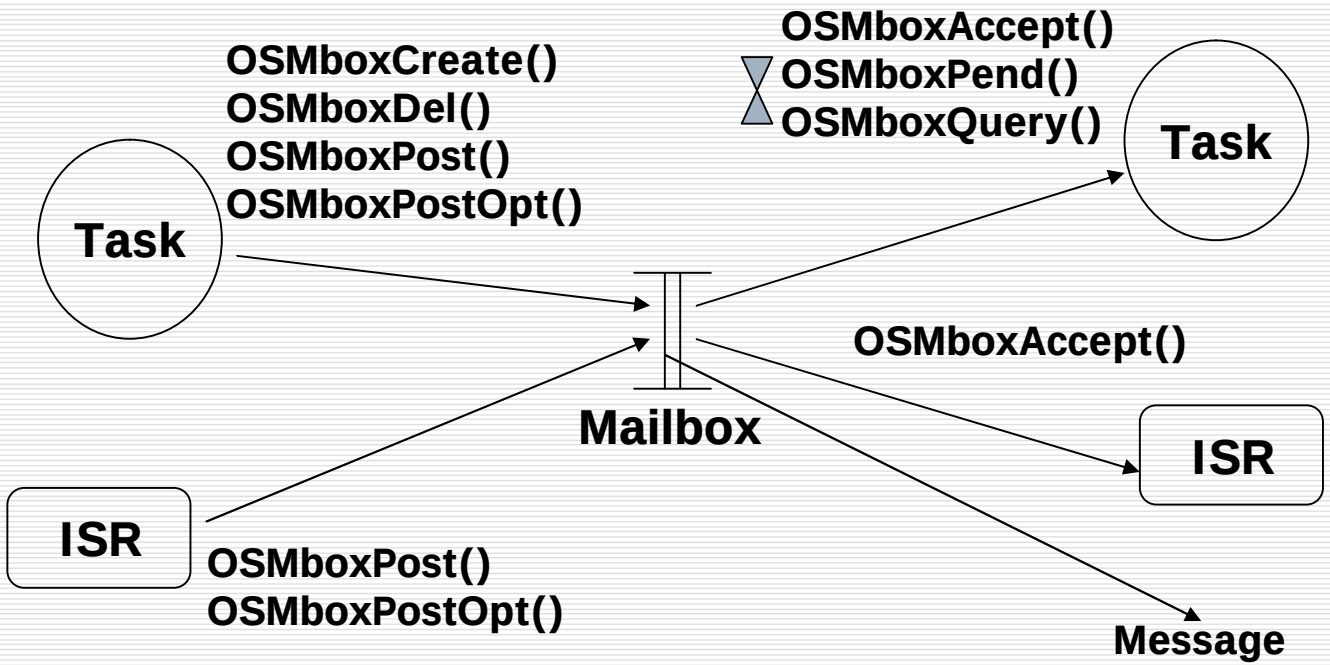
- Introduction
- Creating a Mailbox, OSMboxCreate()
- Deleting a Mailbox, OSMboxDel()
- Waiting for a message at a Mailbox, OSMboxPend()
- Sending a message to a Mailbox, OSMboxPost()
- Sending a message to a Mailbox, OSMboxPostOpt()

Outline (2/2)

- Getting a message without waiting, **OSMboxAccept()**
- Obtaining the status of a Mailbox, **OSMboxQuery()**
- Using a Mailbox as a binary semaphore
- Using a Mailbox instead of OSTimeDly()

Introduction

(1/2)



Introduction

(2/2)

MicroC/OS-II Mailbox Service	Enable when set to 1 in OS_CFG.H
OSMboxAccept()	OS_MBOX_ACCEPT_EN
OSMboxCreate()	
OSMboxDel()	OS_MBOX_DEL_EN
OSMboxPend()	
OSMboxPost()	OS_MBOX_POST_EN
OSMboxPostOpt()	OS_MBOX_POST_OPT_EN
OSMboxQuery()	OS_MBOX_QUERY_EN

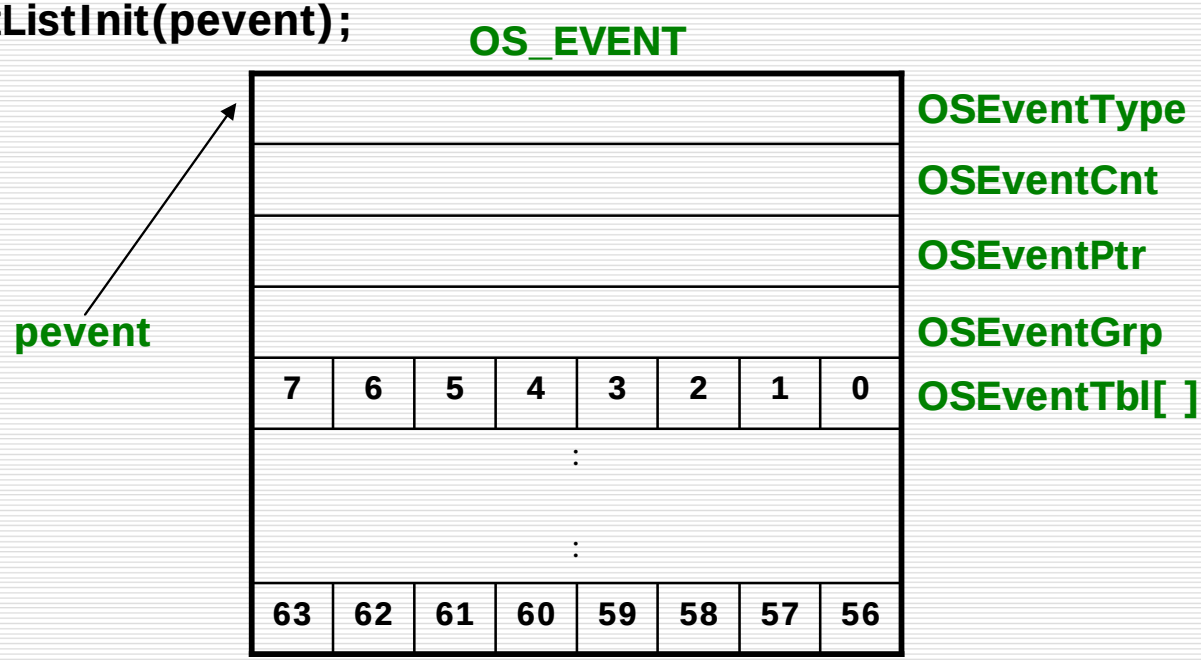
To enable MicroC/OS-II's Mailbox services, you must set the configuration constant OS_MBOX_EN to 1. (OS_CFG.H)

Creating a Mailbox - OSMboxCreate() (1/2)

```
01  OS_EVENT *OSMboxCreate (void *msg)
02  {
03  #if OS_CRITICAL_METHOD == 3
04      OS_CPU_SR cpu_sr;
05  #endif
06      OS_EVENT *pevent;
07      if (OSIntNesting > 0) {
08          return( (OS_EVENT *) 0);
09      }
10      OS_ENTER_CRITICAL();
11      pevent = OSEventFreeList;
12      if (OSEventFreeList != (OS_EVENT *) 0) {
13          OSEventFreeList = (OS_EVENT *) OSEventFreeList->OSEventPtr;
14      }
15      OS_EXIT_CRITICAL();
```

Creating a Mailbox - OSMboxCreate() (2/2)

```
16      if (pevent != (OS_EVENT *) 0) {
17          pevent->OSEventType = OS_EVENT_TYPE_MBOX;
18          pevent->OSEventCnt = 0;
19          pevent->OSEventPtr = msg;
20          OS_EventWaitListInit(pevent);
21      }
22      return(pevent);
23  }
```



Deleting a Mailbox - OSMboxDel() (1/5)

```
01  OS_EVENT *OSMboxDel (OS_EVENT *pevent, INT8U opt, INT8U *err)
02  {
03  #if OS_CRITICAL_METHOD == 3
04      OS_CPU_SR cpu_sr;
05  #endif
06      BOOLEAN tasks_waiting;
07      if (OSIntNesting > 0) {
08          *err = OS_ERR_DEL_ISR;
09          return(pevent);
10      }
11  #if OS_ARG_CHK_EN > 0
12      if (pevent == (OS_EVENT *) 0) {
13          *err = OS_ERR_PEVENT_NULL;
14          return(pevent);
15      }
```

OS_DEL_NO_PEND
OS_DEL_ALWAYS

OS_ARG_CHK_EN → 1. (OS_CFH.H)

Deleting a Mailbox - OSMboxDel() (2/5)

```
16      if (pevent->OSEventType != OS_EVENT_TYPE_MBOX) {
17          *err = OS_ERR_EVENT_TYPE;
18          return(pevent);
19      }
20  #endif
21      OS_ENTER_CRITICAL();
22      if (pevent->OSEventGrp != 0x00) {
23          tasks_waiting = TRUE;
24      } else {
25          tasks_waiting = FALSE;
26      }
```

Deleting a Mailbox - OSMboxDel() (3/5)

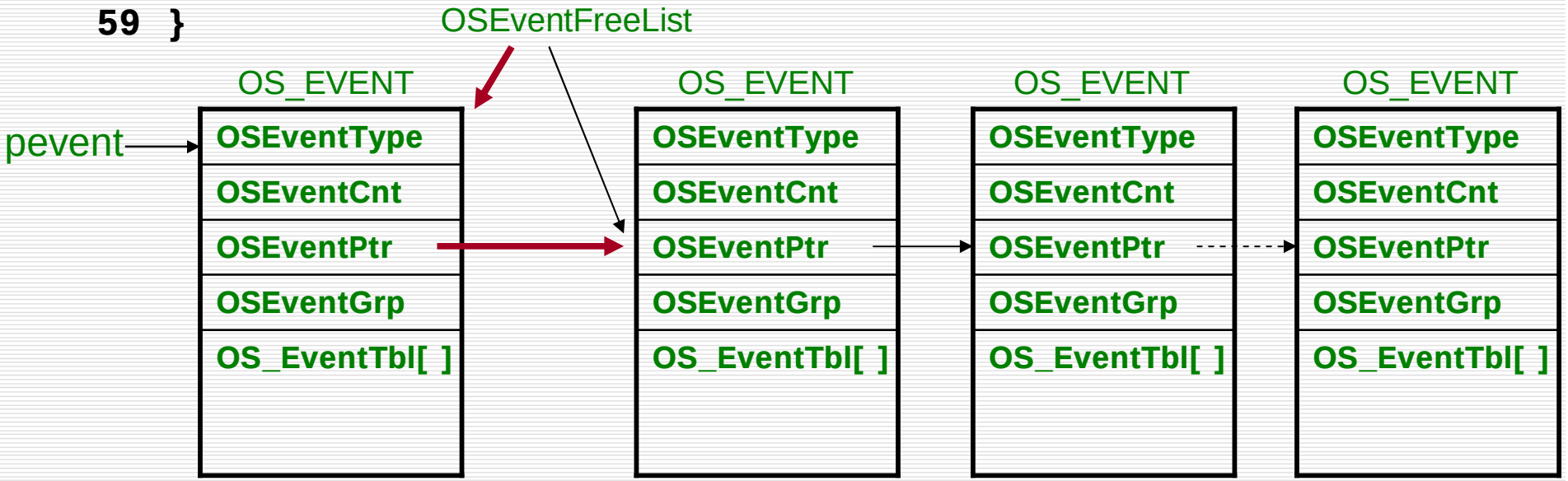
```
27    switch (opt) {
28        case OS_DEL_NO_PEND:
29            if (tasks_waiting == FALSE) {
30                pevent->OSEventType = OS_EVENT_TYPE_UNUSED;
31                pevent->OSEventPtr = OSEventFreeList;
32                OSEventFreeList = pevent;
33                OS_EXIT_CRITICAL();
34                *err = OS_NO_ERR;
35                return( (OS_EVENT *) 0);
36            } else {
37                OS_EXIT_CRITICAL();
38                *err = OS_ERR_TASK_WAITING;
39                return(pevent);
40            }
```

Deleting a Mailbox - OSMboxDel() (4/5)

```
41      case OS_DEL_ALWAYS:
42          while (pevent->OSEventGrp != 0x00) {
43              OS_EventTaskRdy(pevent, (void *) 0, OS_STAT_MBOX);
44          }
45          pevent->OSEventType = OS_EVENT_TYPE_UNUSED;
46          pevent->OS_EventPtr = OSEventFreeList;
47          OSEventFreeList = pevent;
48          OS_EXIT_CRITICAL();
49          if (tasks_waiting == TRUE) {
50              OS_Sched();
51          }
52          *err = OS_NO_ERR:
53          return( (OS_EVENT *) 0);
```

Deleting a Mailbox - OSMboxDel() (5/5)

```
54          default:
55              OS_EXIT_CRITICAL();
56              *err = OS_ERR_INVALID_OPT;
57              return(pevent);
58      }
59  }
```



Waiting for a message at a Mailbox - OSMboxPend() (1/3)

```
01  void *OSMboxPend (OS_EVENT *pevent, INT16U timeout, INT8U *err)
02  {
03  #if OS_CRITICAL_METHOD == 3
04      OS_CPU_SR cpu_sr;
05  #endif
06      void *msg;
07      if (OSIntNesting > 0) {
08          *err = OS_ERR_PEND_ISR;
09          return( (void *) 0);
10      }
11  #if OS_ARG_CHK_EN > 0
12      if (pevent == (OS_EVENT *) 0) {
13          *err = OS_ERR_PEVENT_NULL;
14          return( (void *) 0);
15      }
```

0
<=65535 ticks

Waiting for a message at a Mailbox - OSMboxPend() (2/3)

```
16      if (pevent->OSEventType != OS_EVENT_TYPE_MBOX) {
17          *err = OS_ERR_EVENT_TYPE;
18          return( (void *) 0);
19      }
20  #endif
21      OS_ENTER_CRITICAL();
22      msg = pevent->OSEventPtr;
23      if (msg != (void *) 0) {
24          pevent->OSEventPtr = (void *) 0;
25          OS_EXIT_CRITICAL();
26          *err = OS_NO_ERR;
27          return(msg);
28      }
29      OSTCBCur->OSTCBStat |= OS_STAT_MBOX;
30      OSTCBCur->OSTCBDly = timeout;
31      OS_EventTaskWait(pevent);
```

OSTimeTick()
→ OSTCBDly--

Waiting for a message at a Mailbox - OSMboxPend() (3/3)

```
32     OS_EXIT_CRITICAL();
33     OS_Sched();
34     OS_ENTER_CRITICAL();
35     msg = OSTCBCur->OSTCBMsg;
36     if (msg != (void *) 0) {
37        OSTCBCur->OSTCBMsg = (void *) 0;
38        OSTCBCUr->OSTCBStat = OS_STAT_RDY;
39        OSTCBCur->OSTCBEventPtr = (OS_EVENT *) 0;
40        OS_EXIT_CRITICAL();
41        *err = OS_NO_ERR;
42        return(msg);
43     }
44     OS_EventTO(pevent);
45     OS_EXIT_CRITICAL();
46     *err = OS_TIMEOUT:
47     return( (void *) 0);
48  }
```

Sending a message to a Mailbox - OSMboxPost() (1/2)

```
01  INT8U OSMboxPost (OS_EVENT *pevent, void *msg)
02  {
03  #if OS_CRITICAL_METHOD == 3
04      OS_CPU_SR cpu_sr;
05  #endif
06  #if OS_ARG_CHK_EN > 0
07      if (pevent == (OS_EVENT *) 0) {
08          return(OS_ERR_PEVENT_NULL);
09      }
10      if (msg == (void *) 0) {
11          return(OS_ERR_POST_NULL_PTR);
12      }
13      if (pevent ->OSEventType != OS_EVENT_TYPE_MBOX) {
14          return(OS_ERR_EVENT_TYPE);
15      }
16  #endif
```

Sending a message to a Mailbox - OSMboxPost() (2/2)

```
17     OS_ENTER_CRITICAL();
18     if (pevent->OSEventGrp != 0x00) {
19         OS_EventTaskRdy(pevent, msg, OS_STAT_MBOX);
20         OS_EXIT_CRITICAL();
21         OS_Sched();
22         return(OS_NO_ERR);
23     }
24     if (pevent->OSEventPtr != (void *) 0) {
25         OS_EXIT_CRITICAL();
26         return(OS_MBOX_FULL);
27     }
28     pevent->OSEventPtr = msg;
29     OS_EXIT_CRITICAL();
30     return(OS_NO_ERR);
31 }
```

Sending a message to a Mailbox - OSMboxPostOpt() (1/3)

```
01  INT8U OSMboxPostOpt (OS_EVENT *pevent, void *msg, INT8U opt)
02  {
03  #if OS_CRITICAL_METHOD == 3
04      OS_CPU_SR cpu_sr;
05  #endif
06  #if OS_ARG_CHK_EN > 0
07      if (pevent == (OS_EVENT *) 0) {
08          return(OS_ERR_PEVENT_NULL);
09      }
10      if (msg == (void *) 0) {
11          return(OS_ERR_POST_NULL_PTR);
12      }
13      if (pevent->OSEventType != OS_EVENT_TYPE_MBOX) {
14          return(OS_ERR_EVENT_TYPE);
15      }
16  #endif
```

OS_POST_OPT_NONE
OS_POST_OPT_BROADCAST

Sending a message to a Mailbox - OSMboxPostOpt() (2/3)

```
17    OS_ENTER_CRITICAL();
18    if (pevent->OSEventGrp != 0x00) {
19        if ( (opt & OS_POST_OPT_BROADCAST) != 0x00) {
20            while (pevent->OSEventGrp != 0x00) {
21                OS_EventTaskRdy(pevent, msg, OS_STAT_MBOX);
22            }
23        } else {
24            OS_EventTaskRdy(pevent, msg, OS_STAT_MBOX);
25        }
26        OS_EXIT_CRITICAL();
27        OS_Sched();
28        return(OS_NO_ERR);
29    }
```

Sending a message to a Mailbox - OSMboxPostOpt() (3/3)

```
30      if (pevent->OSEventPtr != (void *) 0) {
31          OS_EXIT_CRITICAL();
32          return(OS_MBOX_FULL);
33      }
34      pevent->OSEventPtr = msg;
35      OS_EXIT_CRITICAL();
36      return(OS_NO_ERR);
37  }
```

Getting a message without waiting - OSMboxAccept() (1/2)

```
01  void *OSMboxAccept (OS_EVENT *pevent)
02  {
03  #if OS_CRITICAL_METHOD == 3
04      OS_CPU_SR cpu_sr;
05  #endif
06      void *msg;
07  #if OS_ARG_CHK_EN > 0
08      if (pevent == (OS_EVENT *) 0) {
09          return( (void *) 0);
10      }
11      if (pevent->OSEventType != OS_EVENT_TYPE_MBOX) {
12          return( (void *) 0);
13  #endif
```

Getting a message without waiting - OSMboxAccept() (2/2)

```
14      OS_ENTER_CRITICAL();
15      msg = pevent->OSEventPtr;
16      pevent->OSEvent = (void *) 0;
17      OS_EXIT_CRITICAL();
18      return(msg);
19  }
```

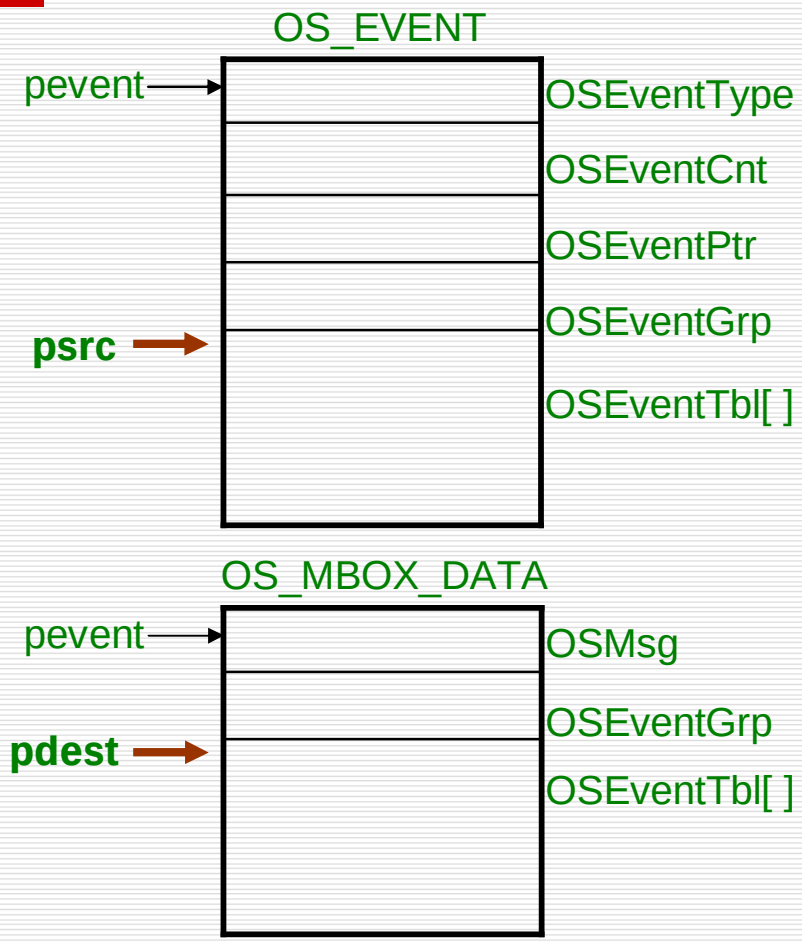
Obtaining the status of a Mailbox - OSMboxQuery() (1/3)

```
01  INT8U OSMboxQuery (OS_EVENT *pevent, OS_MBOX_DATA *pdata)
02  {
03  #if OS_CRITICAL_METHOD == 3
04      OS_CPU_SR cpu_sr;
05  #endif
06      INT8U *psrc;
07      INT8U *pdest;
08  #if OS_ARG_CHK_EN > 0
09      if (pevent == (OS_EVENT *) 0) {
10          return(OS_ERR_PEVENT_NULL);
11      }
12      if (pevent->OSEventType != OS_EVENT_TYPE_MBOX) {
13          return(OS_ERR_EVENT_TYPE);
14      }
15  #endif
16      OS_ENTER_CRITICAL();
17      pdata->OSEventGrp = pevent->OSEventGrp;
```

OSMsg
OSEventGrp
OSEventTbl[ ]

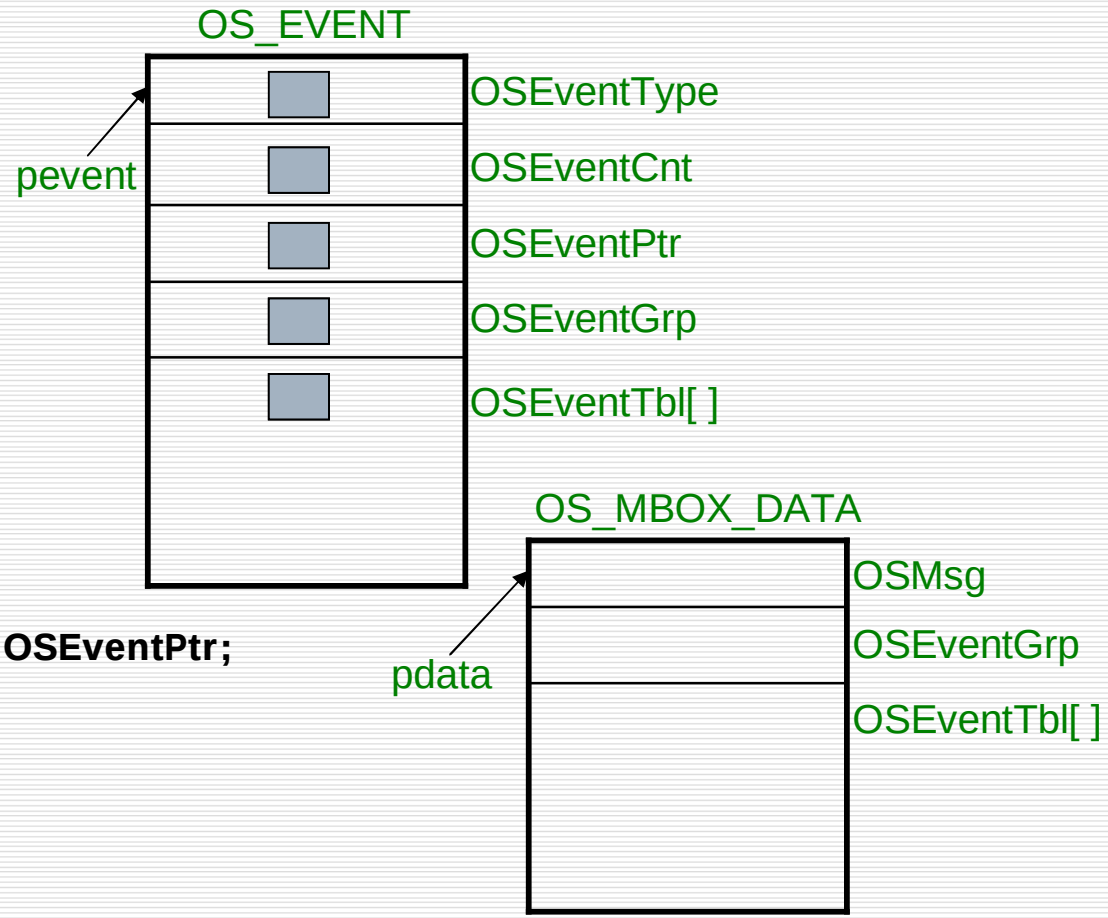
Obtaining the status of a Mailbox - OSMboxQuery() (2/3)

```
18      psrc = &pevent->OSEventTbl[0];
19      pdest = &pdata->OSEventTbl[0];
20  #if OS_EVENT_TBL_SIZE > 0
21      *pdest++ = *psrc++;
22  #endif
23  #if OS_EVENT_TBL_SIZE > 1
24      *pdest++ = *psrc++;
25  #endif
26  #if OS_EVENT_TBL_SIZE > 2
27      *pdest++ = *psrc++;
28  #endif
29  #if OS_EVENT_TBL_SIZE > 3
30      *pdest++ = *psrc++;
31  #endif
```



Obtaining the status of a Mailbox - OSMboxQuery() (3/3)

```
32  #if OS_EVENT_TBL_SIZE > 4
33      *pdest++ = *psrc++;
34  #endif
35  #if OS_EVENT_TBL_SIZE > 5
36      *pdest++ = *psrc++;
37  #endif
38  #if OS_EVENT_TBL_SIZE > 6
39      *pdest++ = *psrc++;
40  #endif
41  #if OS_EVENT_TBL_SIZE > 7
42      #pdest++ = *psrc++;
43  #endif
44      pdata->OSMsg = pevent->OSEventPtr;
45      OS_EXIT_CRITICAL();
46      return(OS_NO_ERR);
47  }
```



Using a Mailbox as a binary semaphore

```
01  OS_EVENT *MboxSem;
02  void Task1 (void *pdata)
03  {
04      INT8U err;
05      for ( ; ;) {
06          OSMboxPend(MboxSem, 0, &err);
07                          :
08          /* Task has semaphore, access resource(s) */
09                          :
10          OSMboxPost(MboxSem, (void *) 1);
11      }
12  }
```

Initializing the mailbox with a non-NULL pointer. (Ex: (void *) 1)

Using a Mailbox instead of OSTimeDly()

```
01 OS_EVENT *MboxTimeDly;
02 void Task1 (void *pdata)
03 {
04     INT8U err;
05     for ( ; ;) {
06         OSMboxPend(MboxTimeDly, TIMEOUT, &err);
07                         :
08         /* Code executed after time delay */
09                         :
10     }
11 void Task2 (void *pdata)
12 {
13     INT8U err;
14     for( ; ;) {
15         OSMboxPost(MboxTimeDly, (void *) 1);
16 }
```

The End