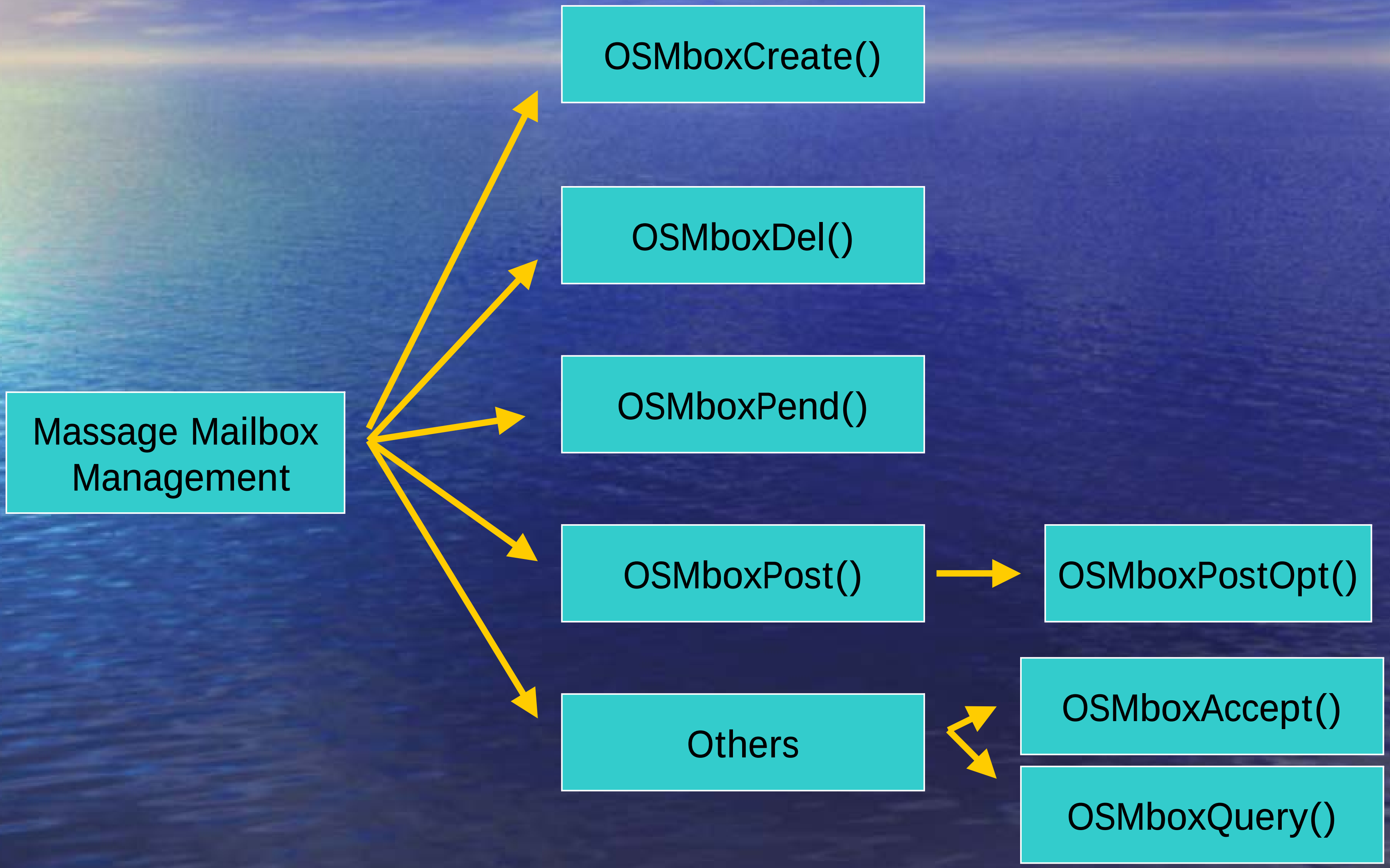
Chapter 10

Message Mailbox Management

Speaker:
Shing-Guo Chang

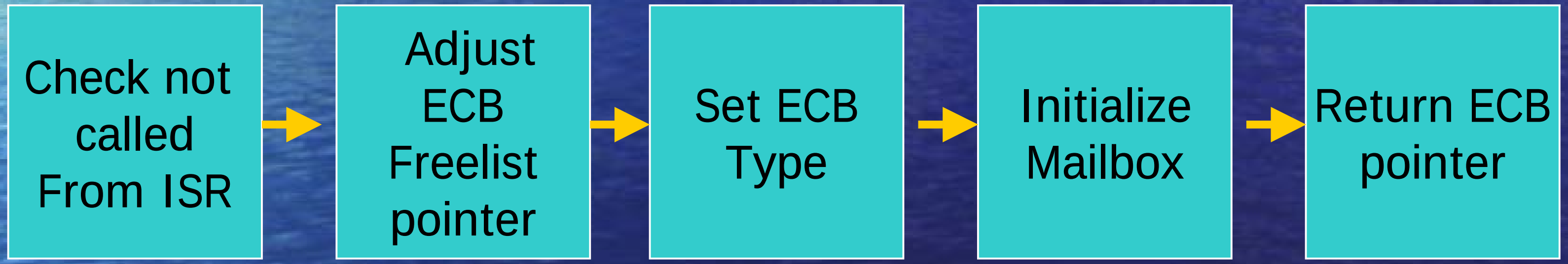
Instruction(1/2)



Instruction(2/2)

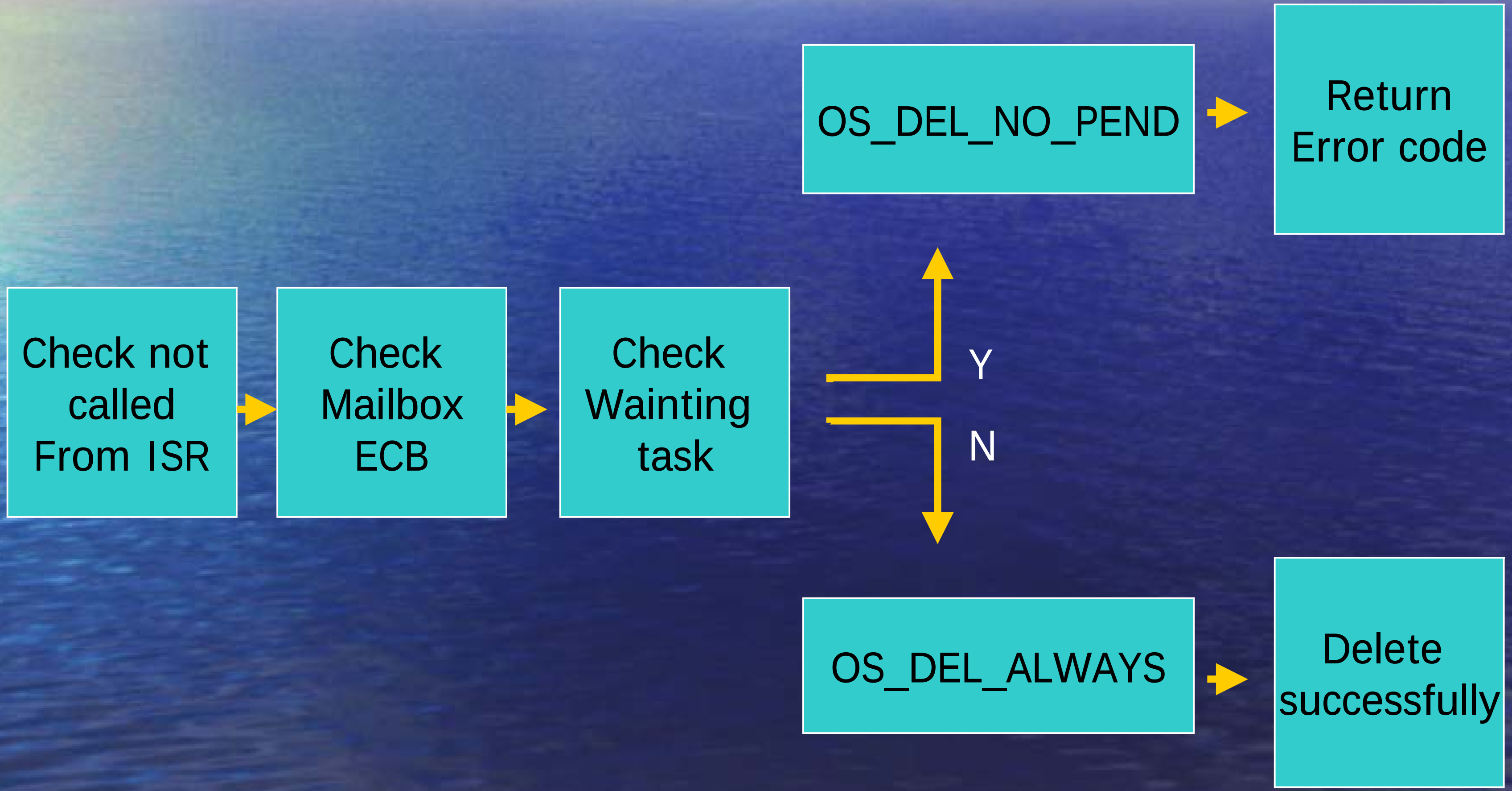
- ISR
  - OSMboxPost()
  - OSMboxPostOpt()
  - OSMboxAccept()

OSMboxCreate()



```
OS_EVENT  *OSMboxCreate (void *msg)
{
#if OS_CRITICAL_METHOD == 3                      /* Allocate storage for CPU status register  */
    OS_CPU_SR  cpu_sr;
#endif
    OS_EVENT  *pevent;


    if (OSIntNesting > 0) {                      /* See if called from ISR ...  */
        return ((OS_EVENT *)0);                  /* ... can't CREATE from an ISR    */
    }
    OS_ENTER_CRITICAL();
    pevent = OSEventFreeList;                    /* Get next free event control block    */
    if (OSEventFreeList != (OS_EVENT *)0) {      /* See if pool of free ECB pool was empty  */
        OSEventFreeList = (OS_EVENT *)OSEventFreeList->OSEventPtr;
    }
    OS_EXIT_CRITICAL();
    if (pevent != (OS_EVENT *)0) {
        pevent->OSEventType = OS_EVENT_TYPE_MBOX;
        pevent->OSEventCnt  = 0;
        pevent->OSEventPtr  = msg;               /* Deposit message in event control block    */
        OS_EventWaitListInit(pevent);
    }
    return (pevent);                             /* Return pointer to event control block           */
}
```

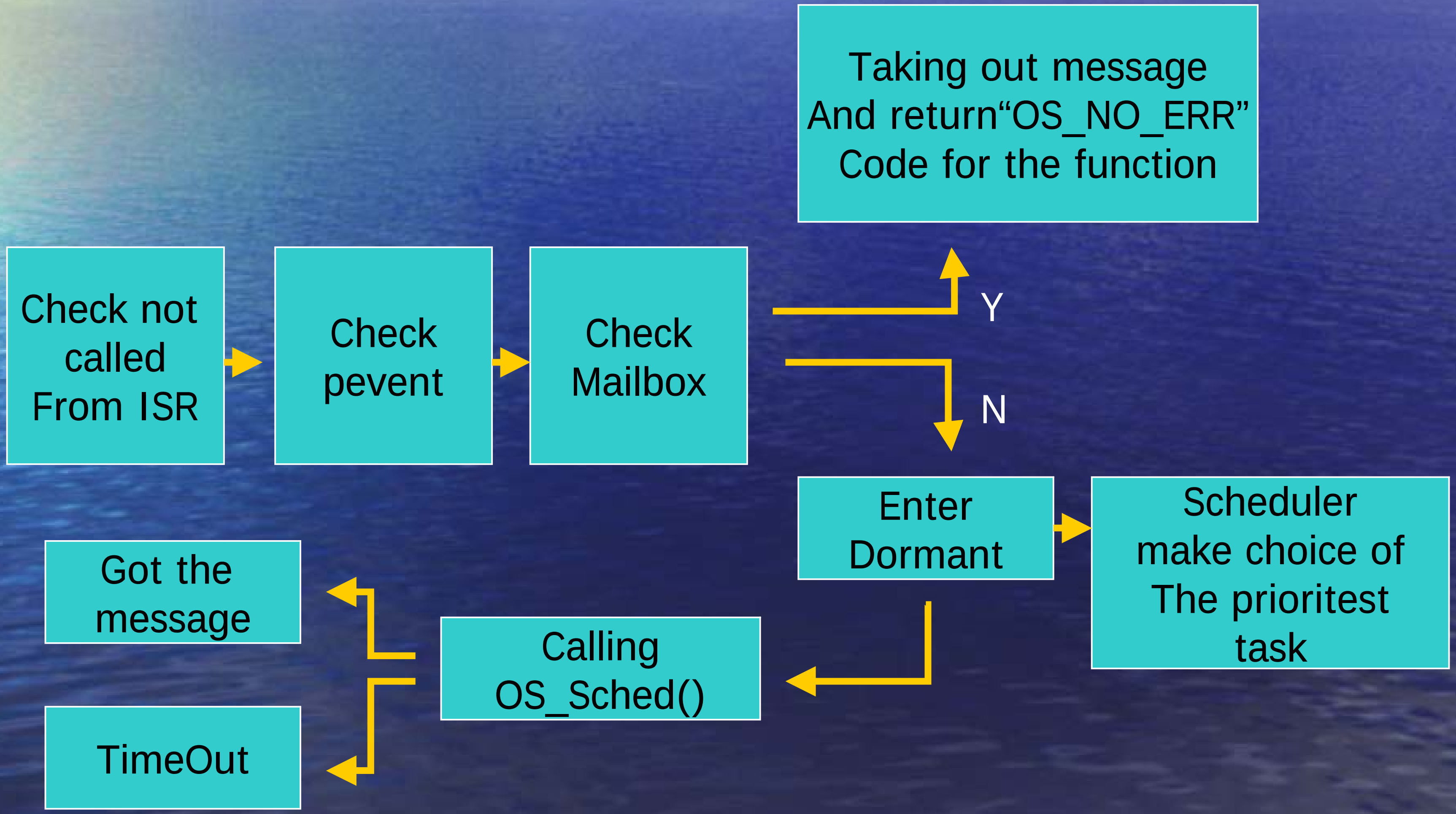
OSMboxDel()


Check not called From ISR
Check Mailbox ECB
Check Wainting task
Y
N
OS_DEL_NO_PEND
Return Error code
OS_DEL_ALWAYS
Delete successfully

```
OS_EVENT  *OSMboxDel (OS_EVENT *pevent, INT8U opt, INT8U *err){
#if OS_CRITICAL_METHOD == 3                        /* Allocate storage for CPU status register */
    OS_CPU_SR  cpu_sr;
#endif
    BOOLEAN    tasks_waiting;


    if (OSIntNesting > 0) {                        /* See if called from ISR ...               */
        *err = OS_ERR_DEL_ISR;                     /* ... can't DELETE from an ISR             */
        return (pevent);
    }
#if OS_ARG_CHK_EN > 0
    if (pevent == (OS_EVENT *)0) {                 /* Validate 'pevent'                        */
        *err = OS_ERR_PEVENT_NULL;
        return (pevent);
    }
    if (pevent->OSEventType != OS_EVENT_TYPE_MBOX) {   /* Validate event block type   */
        *err = OS_ERR_EVENT_TYPE;
        return (pevent);
    }
#endif
    OS_ENTER_CRITICAL();
    if (pevent->OSEventGrp != 0x00) {              /* See if any tasks waiting on mailbox  */
        tasks_waiting = TRUE;                      /* Yes                                  */
    } else {
        tasks_waiting = FALSE;                     /* No                                   */
    }
```

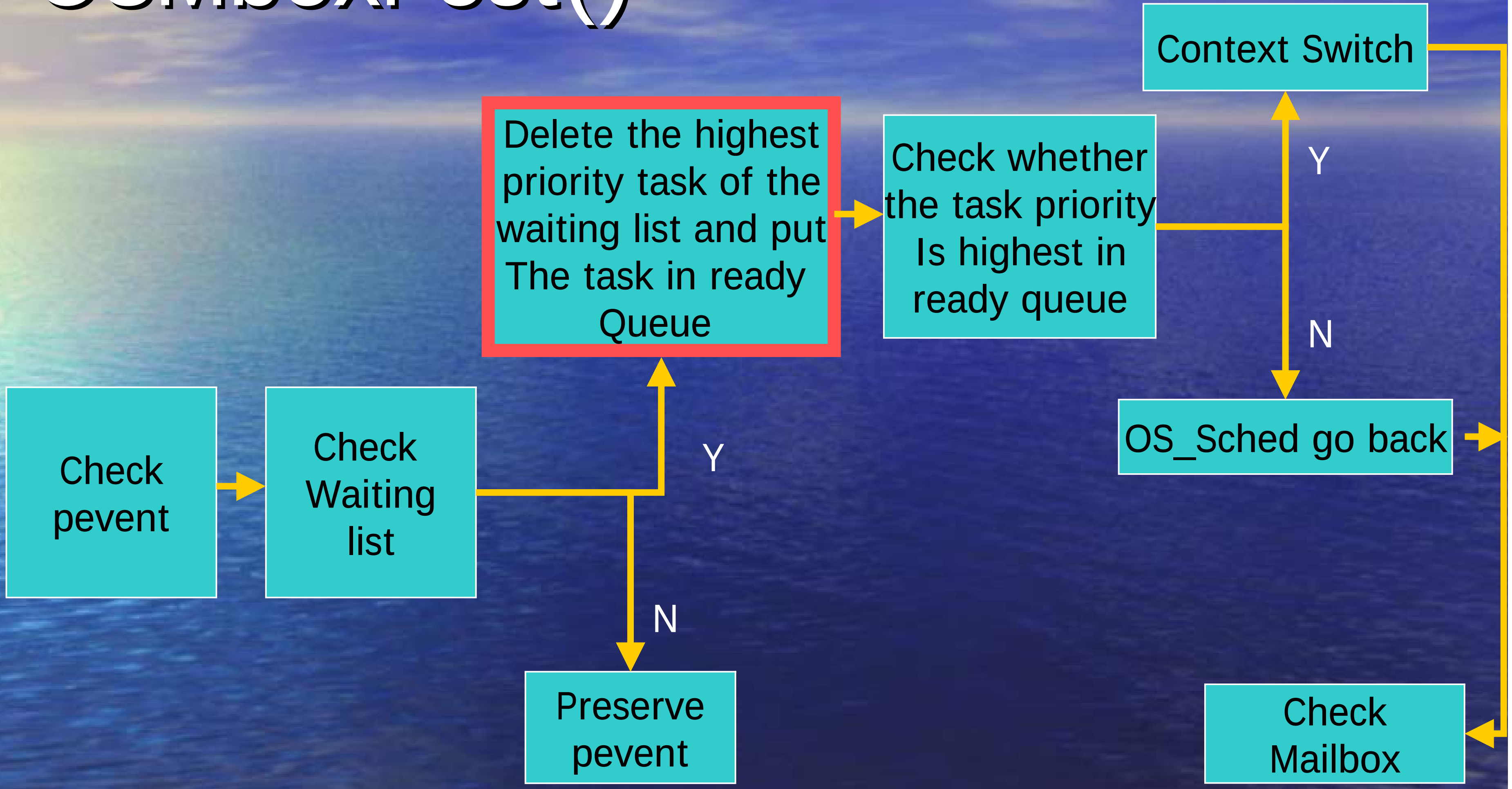
```
switch (opt) {
        case OS_DEL_NO_PEND:                      /* Delete mailbox only if no task waiting   */
             if (tasks_waiting == FALSE) {
                 pevent->OSEventType = OS_EVENT_TYPE_UNUSED;
                 pevent->OSEventPtr  = OSEventFreeList;
                                                          /* Return Event Control Block to free list*/
                 OSEventFreeList     = pevent;            /* Get next free event control block        */
                 OS_EXIT_CRITICAL();
                 *err = OS_NO_ERR;
                 return ((OS_EVENT *)0);                  /* Mailbox has been deleted                 */
             } else {
                 OS_EXIT_CRITICAL();
                 *err = OS_ERR_TASK_WAITING;
                 return (pevent);
             }
case OS_DEL_ALWAYS:                                         /* Always delete the mailbox             */
         while (pevent->OSEventGrp != 0x00) {
                                                            /* Ready ALL tasks waiting for mailbox*/
            OS_EventTaskRdy(pevent, (void *)0, OS_STAT_MBOX);
         }
         pevent->OSEventType = OS_EVENT_TYPE_UNUSED;
         pevent->OSEventPtr  = OSEventFreeList;
                                                            /* Return Event Control Block to free list*/
         OSEventFreeList     = pevent;                      /* Get next free event control block      */
         OS_EXIT_CRITICAL();
         if (tasks_waiting == TRUE) {                   /* Reschedule only if task(s) were waiting  */
            OS_Sched();                                 /* Find highest priority task ready to run  */
         }
         *err = OS_NO_ERR;
         return ((OS_EVENT *)0);                            /* Mailbox has been deleted              */

     default:
         OS_EXIT_CRITICAL();
         *err = OS_ERR_INVALID_OPT;
         return (pevent);
   }
}
#endif
```

OSMboxPend()


Taking out message
And return“OS_NO_ERR”
Code for the function
Check not called From ISR
Check pevent
Check Mailbox
Y
N
Enter Dormant
Scheduler make choice of The prioritest task
Got the message
Calling OS_Sched()
TimeOut

```
void  *OSMboxPend (OS_EVENT *pevent, INT16U timeout, INT8U *err)
{
#if OS_CRITICAL_METHOD == 3                      /* Allocate storage for CPU status register */
    OS_CPU_SR  cpu_sr;
#endif
    void      *msg;


    if (OSIntNesting > 0) {                      /* See if called from ISR ...               */
        *err = OS_ERR_PEND_ISR;                  /* ... can't PEND from an ISR               */
        return ((void *)0);
    }
#if OS_ARG_CHK_EN > 0
    if (pevent == (OS_EVENT *)0) {               /* Validate 'pevent'                        */
        *err = OS_ERR_PEVENT_NULL;
        return ((void *)0);
    }
    if (pevent->OSEventType != OS_EVENT_TYPE_MBOX) {  /* Validate event block type */
        *err = OS_ERR_EVENT_TYPE;
        return ((void *)0);
    }
#endif
OS_ENTER_CRITICAL();
    msg = pevent->OSEventPtr;
    if (msg != (void *)0) {                          /* See if there is already a message              */
        pevent->OSEventPtr = (void *)0;               /* Clear the mailbox                              */
        OS_EXIT_CRITICAL();
        *err = OS_NO_ERR;
        return (msg);                                 /* Return the message received (or NULL)          */
    }
    OSTCBCur->OSTCBStat |= OS_STAT_MBOX;              /* Message not available, task will pend */
    OSTCBCur->OSTCBDly    = timeout;                  /* Load timeout in TCB                            */
    OS_EventTaskWait(pevent);                         /* Suspend task until event or timeout occurs*/
    OS_EXIT_CRITICAL();
    OS_Sched();                                       /* Find next highest priority task ready to run   */
    OS_ENTER_CRITICAL();
    msg = OSTCBCur->OSTCBMsg;
    if (msg != (void *)0) {                           /* See if we were given the message               */
        OSTCBCur->OSTCBMsg       = (void *)0;         /* Yes, clear message received                    */
        OSTCBCur->OSTCBStat      = OS_STAT_RDY;
        OSTCBCur->OSTCBEventPtr = (OS_EVENT *)0;      /* No longer waiting for event                    */
        OS_EXIT_CRITICAL();
        *err                    = OS_NO_ERR;
        return (msg);                                 /* Return the message received                    */
    }
    OS_EventTO(pevent);                               /* Timed out, Make task ready                     */
    OS_EXIT_CRITICAL();
    *err = OS_TIMEOUT;                                /* Indicate that a timeout occured                */
    return ((void *)0);                               /* Return a NULL message                          */
}
```

OSMboxPost()


Check pevent
Check Waiting list
Y
Delete the highest priority task of the waiting list and put The task in ready Queue
N
Preserve pevent
Check whether the task priority Is highest in ready queue
Y
Context Switch
N
OS_Sched go back
Check Mailbox

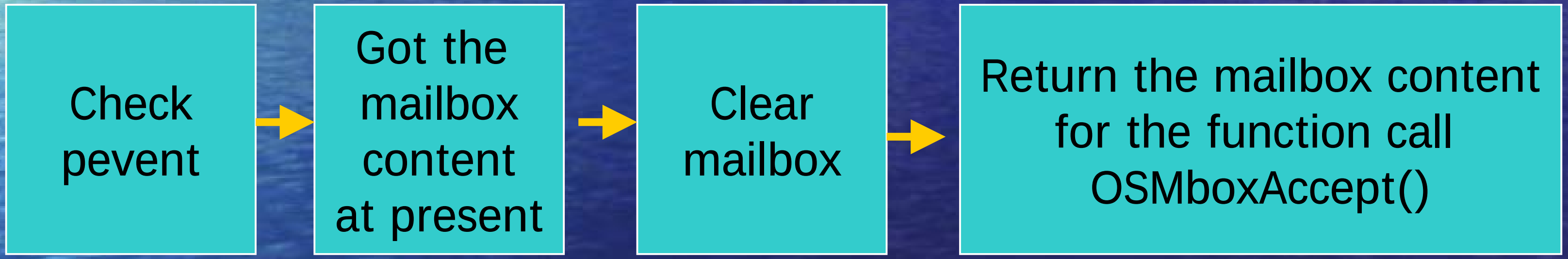
```
#if OS_MBOX_POST_EN > 0
INT8U  OSMboxPost (OS_EVENT *pevent, void *msg)
{
#if OS_CRITICAL_METHOD == 3                      /* Allocate storage for CPU status register*/
    OS_CPU_SR  cpu_sr;
#endif


#if OS_ARG_CHK_EN > 0
    if (pevent == (OS_EVENT *)0) {                /* Validate 'pevent'                         */
        return (OS_ERR_PEVENT_NULL);
    }
    if (msg == (void *)0) {                       /* Make sure we are not posting a NULL pointer*/
        return (OS_ERR_POST_NULL_PTR);
    }
    if (pevent->OSEventType != OS_EVENT_TYPE_MBOX) {  /* Validate event block type      */
        return (OS_ERR_EVENT_TYPE);
    }
#endif
OS_ENTER_CRITICAL();
    if (pevent->OSEventGrp != 0x00) {                   /* See if any task pending on mailbox      */
        OS_EventTaskRdy(pevent, msg, OS_STAT_MBOX);
                                                     /* Ready highest priority task waiting on event   */
        OS_EXIT_CRITICAL();
        OS_Sched();                                  /* Find highest priority task ready to run       */
        return (OS_NO_ERR);
    }
    if (pevent->OSEventPtr != (void *)0) {
                                                     /* Make sure mailbox doesn't already have a msg  */
        OS_EXIT_CRITICAL();
        return (OS_MBOX_FULL);
    }
    pevent->OSEventPtr = msg;                          /* Place message in mailbox                   */
    OS_EXIT_CRITICAL();
    return (OS_NO_ERR);
}
#endif
```

OSMboxPostOpt()

- Opt parameter
  - OS_POST_OPT_BROADCAST

```
OS_ENTER_CRITICAL();
    if (pevent->OSEventGrp != 0x00) {                  /* See if any task pending on mailbox */
        if ((opt & OS_POST_OPT_BROADCAST) != 0x00) {
                                        /* Do we need to post msg to ALL waiting tasks ? */
            while (pevent->OSEventGrp != 0x00) {
                                        /* Yes, Post to ALL tasks waiting on mailbox     */
                OS_EventTaskRdy(pevent, msg, OS_STAT_MBOX);
            }
        } else {
            OS_EventTaskRdy(pevent, msg, OS_STAT_MBOX);
                                        /* No,  Post to HPT waiting on mbox         */
        }
        OS_EXIT_CRITICAL();
        OS_Sched();                                    /* Find highest priority task ready to run  */
        return (OS_NO_ERR);
    }
    if (pevent->OSEventPtr != (void *)0) {
                                        /* Make sure mailbox doesn't already have a msg  */
        OS_EXIT_CRITICAL();
        return (OS_MBOX_FULL);
    }
    pevent->OSEventPtr = msg;                          /* Place message in mailbox              */
    OS_EXIT_CRITICAL();
    return (OS_NO_ERR);
}
#endif
```

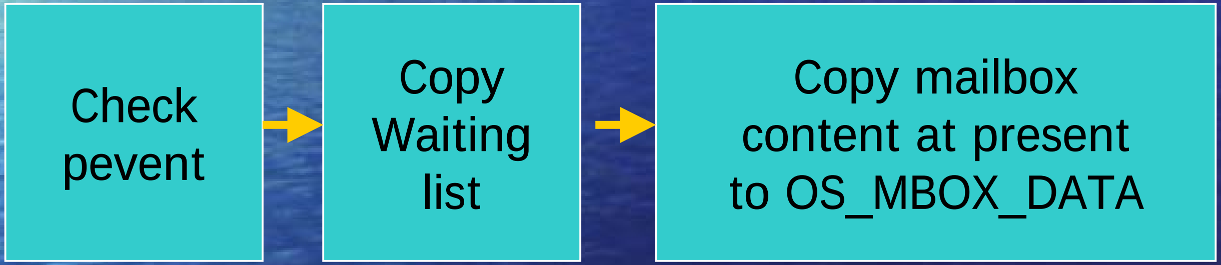
OSMboxAccept()


Check pevent
Got the mailbox content at present
Clear mailbox
Return the mailbox content for the function call OSMboxAccept()

*ISR always get message by the OSMboxAccept()

* Clearing Mailbox is always uesd by OSMboxAccept()

OSMboxQuery()



Other

- Using a Mailbox as a Binary Semaphore
  - OSMboxPend();
  - OSMboxPost();
- Using a Mailbox Instead os OSTimeDly()

Thank you