Message Queue Management

中興大學資訊科學系
指導教授：張軒彬
學生：陳奇進

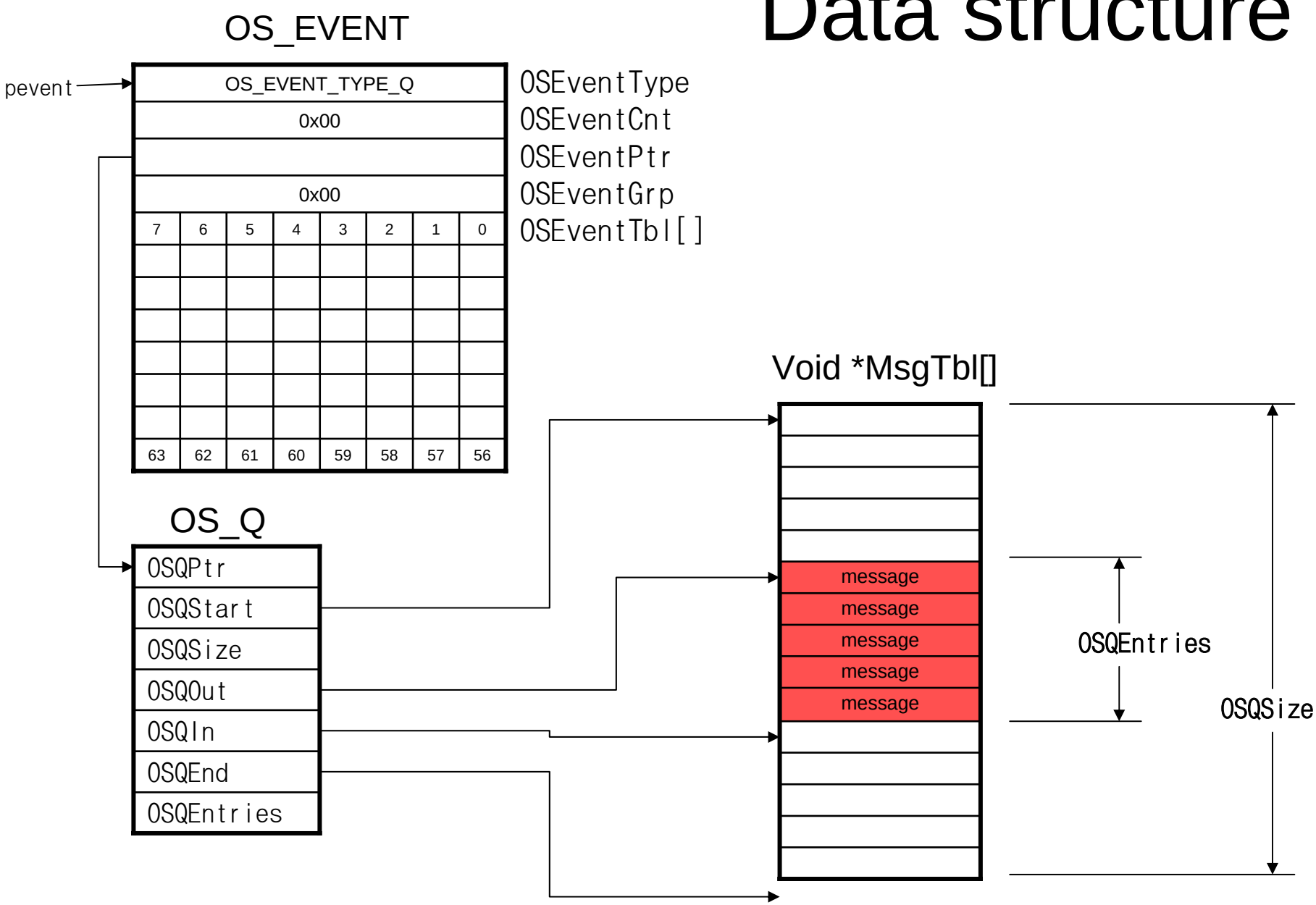
Message Queue Management

- Introduction
- Data structure
- List of free queue control blocks
- Circular buffer of pointers
- µC/OS-II provides nine services
  - OSQCreate()
  - OSQDel()
  - OSQFlush()
  - OSQAccept()
  - OSQQuery()
  - OSQPost()
  - OSQPostOpt()
  - OSQPostFront()
  - OSQPend()

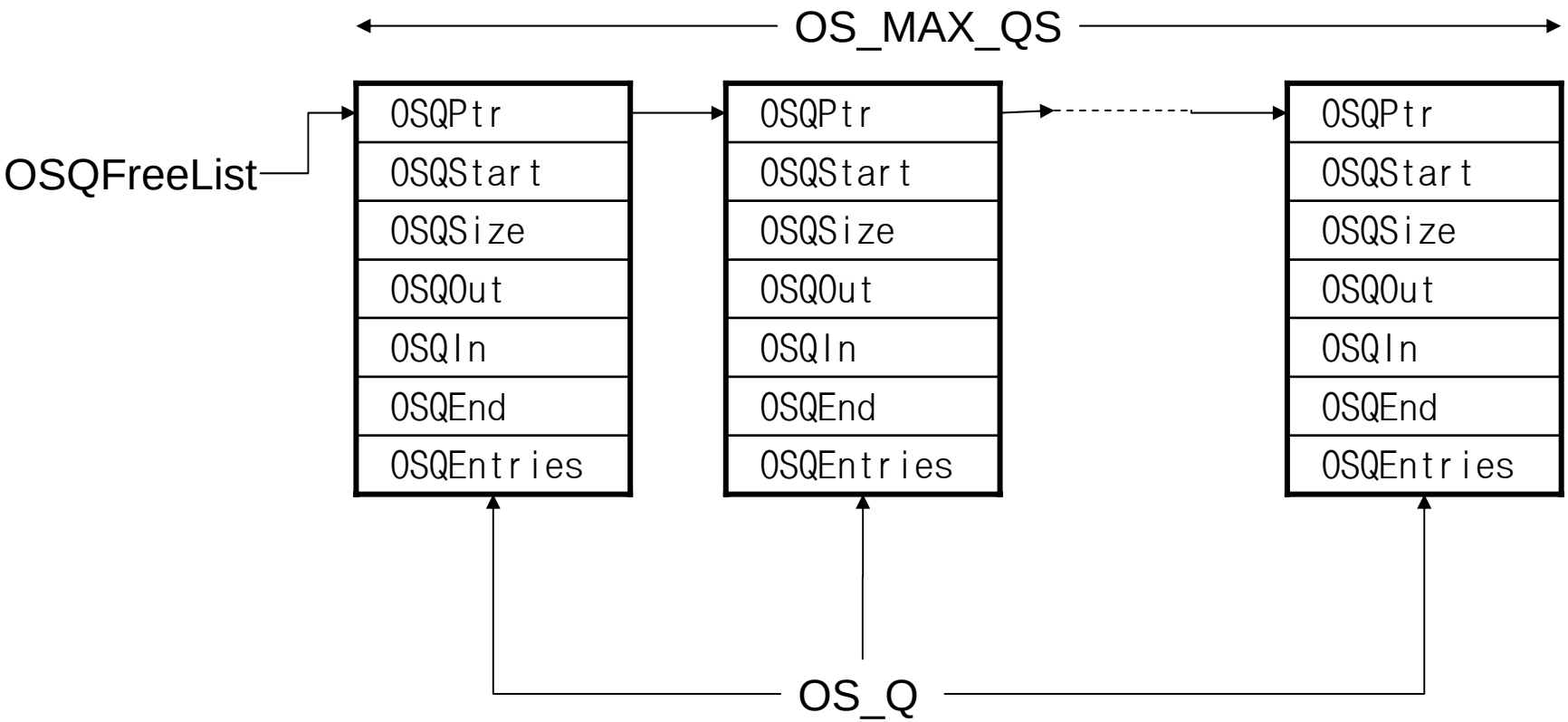
Introduction

- A message queue is a µC/OS-II object that allows a task or an ISR to send a pointer-sized variable to another task.
- The msg pointer would typically be initialized to point to some application specific data structure containing a message
- A queue looks like a mailbox with multiple entries.

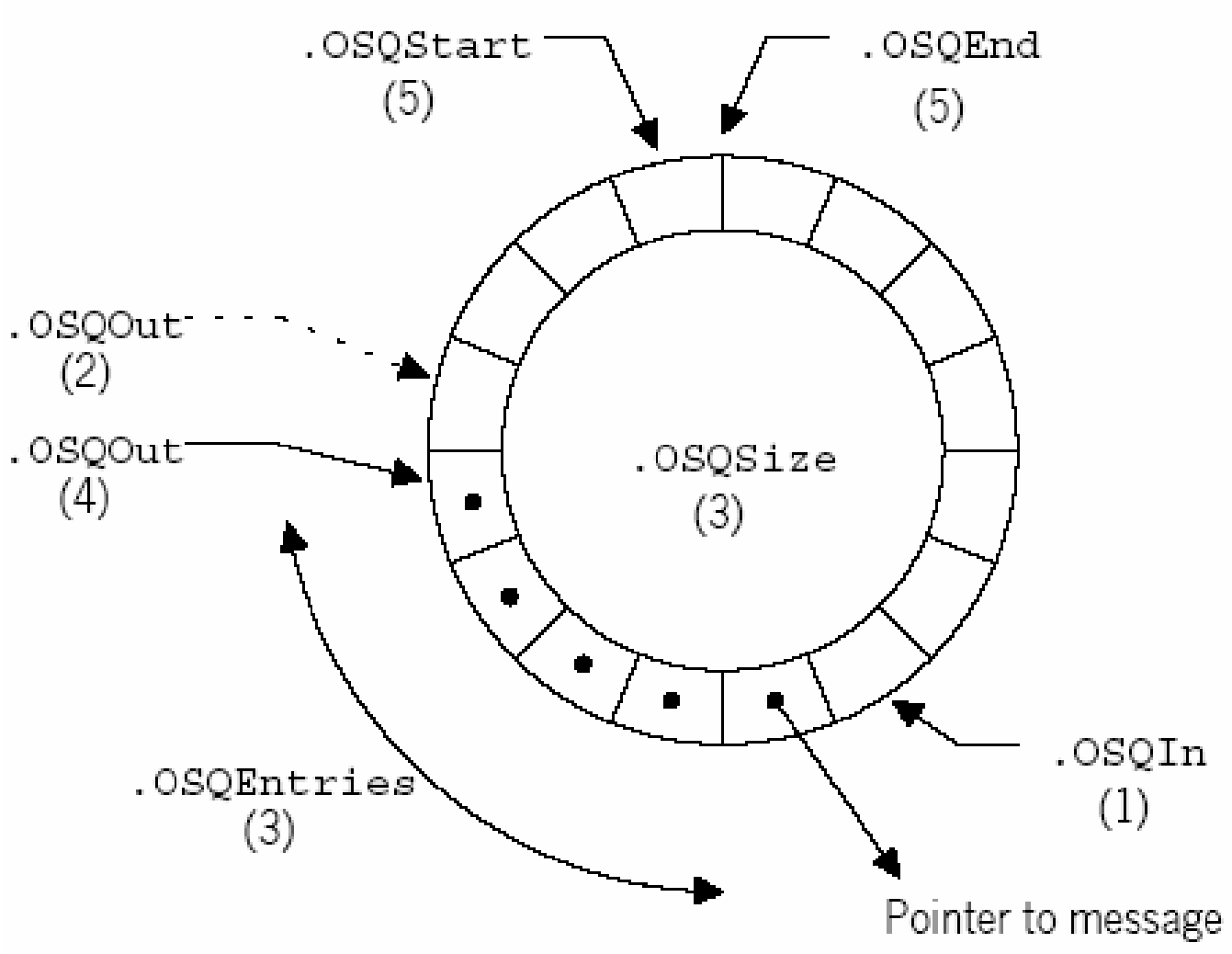
Data structure



List of free queue control blocks



Circular buffer of pointers



OSQCreate()

- **Prototype**：`OS_EVENT OSQCreate(void **start, INT16U size)`
- **Description**： This function creates a message queue if free ECB are available.
- **Arguments**：
  - Start　is a pointer to the base address of the message queue storage area. The storage area MUST be declared as an array of pointers to 'void' as follows:
    `void *MessageStorage[size]`
  - Size　is the number of elements in the storage area
- **Returns**：
  - `!= (OS_EVENT *)0`　is a pointer to the event control clock.
  - `== (OS_EVENT *)0`　if no event control blocks were available or error.

OSQCreate()

```
OS_EVENT  *pevent;
OS_Q      *pq;

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;

if (pevent != (OS_EVENT *)0)                    // See if we have an event control block
{
    pq = OSQFreeList;                           //Get a free queue control block
    if (pq != (OS_Q *)0)                        //Were we able to get a queue control block ?
    {
      OSQFreeList         = OSQFreeList->OSQPtr;
      pq->OSQStart        = start;                     //Initialize the queue
      pq->OSQEnd          = &start[size];
      pq->OSQIn           = start;
      pq->OSQOut          = start;
      pq->OSQSize         = size;
      pq->OSQEntries      = 0;
      pevent->OSEventType = OS_EVENT_TYPE_Q;
      pevent->OSEventCnt  = 0;
      pevent->OSEventPtr  = pq;
      OS_EventWaitListInit(pevent);                //Initalize the wait list
    }
```

OSQCreate()

```
    else                                    //there is no free ECB
    {
        pevent->OSEventPtr = (void *)OSEventFreeList;
        OSEventFreeList    = pevent;
        pevent = (OS_EVENT *)0;
     }
}
return (pevent);
```

OSQDel()

- **Prototype：** `OS_EVENT *OSQDel (OS_EVENT *pevent, INT8U opt, INT8U *err)`
- **Description：** This function deletes a queue and readies all tasks pending on the queue.
- **Arguments：**
  - pevent is a pointer to the ECB.
  - opt determines delete options as follows
    - `OS_DEL_NO_PEND` Delete the mailbox ONLY if no task pending.
    - `OS_DEL_ALWAYS` Deletes the mailbox even if tasks are waiting.

OSQDel()

- **Arguments：**
  - err　is a pointer to an error code that can contain one of the following values:
    - OS_NO_ERR
    - OS_ERR_DEL_ISR
    - OS_ERR_INVALID_OPT
    - OS_ERR_TASK_WAITING
    - OS_ERR_EVENT_TYPE
    - OS_ERR_PEVENT_NULL
- **Returns：**
  - peven　upon error
  - (OS_EVENT *)0　if the queue was successfully deleted.

OSQDel()

```
BOOLEAN    tasks_waiting;
OS_Q                    *pq;


if (pevent->OSEventGrp != 0x00)                    //See if any tasks waiting on queue
    tasks_waiting = TRUE;
else
    tasks_waiting = FALSE;
```

OSQDel()

```
switch (opt)
{
      case OS_DEL_NO_PEND:                                  //Delete queue only if no task waiting
         if (tasks_waiting == FALSE)
         {
           pq                  = (OS_Q *)pevent->OSEventPtr;
           pq->OSQPtr          = OSQFreeList;
           OSQFreeList         = pq;
           pevent->OSEventType = OS_EVENT_TYPE_UNUSED;
           pevent->OSEventPtr  = OSEventFreeList;   //Return Event Control Block to free list
           OSEventFreeList     = pevent;            //Get next free event control block

           *err = OS_NO_ERR;
           return ((OS_EVENT *)0);                  //Queue has been deleted
         }

         else
         {
           *err = OS_ERR_TASK_WAITING;
           return (pevent);
         }
```

OSQDel()

```
    case OS_DEL_ALWAYS:                     //Always delete the queue
            while (pevent->OSEventGrp != 0x00)
                 OS_EventTaskRdy(pevent, (void *)0, OS_STAT_Q);

            pq                  = (OS_Q *)pevent->OSEventPtr;
            pq->OSQPtr          = OSQFreeList;
            OSQFreeList         = pq;
            pevent->OSEventType = OS_EVENT_TYPE_UNUSED;
            pevent->OSEventPtr  = OSEventFreeList;
            OSEventFreeList     = pevent;

            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);         //Queue has been deleted

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

OSQFlush()

- **Prototype**：INT8U OSQFlush (OS_EVENT *pevent)
- **Description**： This function is used to flush the contents of the message queue.
- **Arguments**：
  - pevent　is a pointer to the ECB.
- Return：
  - OS_NO_ERR
  - OS_ERR_EVENT_TYPE
  - OS_ERR_PEVENT_NULL

OSQFlush()

```
OS_Q      *pq;

pq             = (OS_Q *)pevent->OSEventPtr;  //Point to queue storage structure
pq->OSQIn      = pq->OSQStart;
pq->OSQOut     = pq->OSQStart;
pq->OSQEntries = 0;
return (OS_NO_ERR);
```

OSQPend()

- **Prototype**：`void *OSQPend (OS_EVENT *pevent, INT16U timeout INT8U *err)`
- **Description**： This function waits for a message to be sent to a queue.
- **Arguments**：
  - `pevent` is a pointer to the ECB.
  - `timeout` is an optional timeout period (in clock ticks).
  - `err` is a pointer to where an error message will be deposited.
    - `OS_NO_ERR`
    - `OS_TIMEOUT`
    - `OS_ERR_EVENT_TYPE`
    - `OS_ERR_PEND_ISR`
    - `OS_ERR_PEVENT_NULL`

OSQPend()

- **Returns**：
  - `!= (void *)0` is a pointer to the message received
  - `== (void *)0`
    - if no message was received
    - if 'pevent' is a NULL pointer
    - if you didn't pass a pointer to a queue.

OSQPend()

```
pq = (OS_Q *)pevent->OSEventPtr;          //Point at queue control block
if (pq->OSQEntries > 0)
{                                         //See if any messages in the queue
    msg    = *pq->OSQOut++;               //Yes, extract oldest message from the queue
    pq->OSQEntries--;                     //Update the number  of entries in the queue
    if (pq->OSQOut == pq->OSQEnd)         //Wrap OUT pointer if we are at the end of the
        pq->OSQOut = pq->OSQStart;        //queue

    *err = OS_NO_ERR;
    return (msg);                         //Return message received
}
OSTCBCur->OSTCBStat  |= OS_STAT_Q;
OSTCBCur->OSTCBDly    = timeout;          //Load timeout into  TCB
OS_EventTaskWait(pevent);                 //Suspend task until event or timeout occurs

OS_Sched();                               //Find next highest priority task ready to run
```

OSQPend()

```
msg = OSTCBCur->OSTCBMsg;
if (msg    != (void *)0)                        // Did we get a message?
{
    OSTCBCur->OSTCBMsg          = (void *)0;     //Extract message from TCB
    OSTCBCur->OSTCBStat         = OS_STAT_RDY;
    OSTCBCur->OSTCBEventPtr     = (OS_EVENT *)0; //No longer waiting for event
    *err                        = OS_NO_ERR;
    return (msg);                                //Return message received
}
OS_EventTO(pevent);                              //Timed out
*err = OS_TIMEOUT;                               //Indicate a timeout occured
return ((void *)0);                              //No message received
```

OSQPost ()

- **Prototype**：`INT8U  OSQPost (OS_EVENT *pevent, void *msg)`
- **Description**：This function sends a message to a queue.
- **Arguments**：
  - `pevent`　is a pointer to the event control block associated with the desired queue
  - `msg`　is a pointer to the message to send.
- **Returns**：
  - `OS_NO_ERR`
  - `OS_Q_FULL`
  - `OS_ERR_EVENT_TYPE`
  - `OS_ERR_PEVENT_NULL`
  - `OS_ERR_POST_NULL_PTR`

OSQPost()

```
OS_Q      *pq;
if (pevent->OSEventGrp != 0x00)          //See if any task pending on queue
{
    OS_EventTaskRdy(pevent, msg, OS_STAT_Q);
    OS_Sched();                          //Find highest priority task ready to run
    return (OS_NO_ERR);
}
pq = (OS_Q *)pevent->OSEventPtr;         //Point to queue control block
if (pq->OSQEntries >= pq->OSQSize)       //Make sure queue is not full
    return (OS_Q_FULL);

*pq->OSQIn++ = msg;                      //Insert message into queue
pq->OSQEntries++;                        //Update the nbr of entries in the queue
if (pq->OSQIn == pq->OSQEnd)             //Wrap IN ptr if we are at end of queue
    pq->OSQIn = pq->OSQStart;
return (OS_NO_ERR);
```

OSQPostFront ()

- **Prototype**： INT8U OSQPost (OS_EVENT *pevent,
  void *msg)
- **Description**：This function is the same as OSQPost() except that the message is posted at the front instead of the end of the queue.
- **Arguments**：the same as OSQPost()
- **Returns**： the same as OSQPost()

OSQPostFront ()

```
OS_Q      *pq;
if (pevent->OSEventGrp != 0x00)           //See if any task pending on queue
{
    OS_EventTaskRdy(pevent, msg, OS_STAT_Q);
    OS_Sched();                           //Find highest priority task ready to run
    return (OS_NO_ERR);
}
pq = (OS_Q *)pevent->OSEventPtr;          //Point to queue control block
if (pq->OSQEntries >= pq->OSQSize)        //Make sure queue is not full
    return (OS_Q_FULL);
if (pq->OSQOut == pq->OSQStart)           //Wrap OUT ptr if we are at the 1st queue entry
    pq->OSQOut = pq->OSQEnd;
pq->OSQOut--;
*pq->OSQOut = msg;                        //Insert message into queue
pq->OSQEntries++;                         //Update the nbr of entries in the queue
return (OS_NO_ERR);
```

OSQPostOpt ()

- **Prototype**：`INT8U  OSQPostOpt (OS_EVENT *pevent, void *msg, INT8U opt)`
- **Description**：Extended OSQPost().
- **Arguments**：
  - `pevent` is a pointer to the ECB associated with the desired queue.
  - `msg` is a pointer to the message to send.
  - `opt` determines the type of POST performed:
    - `OS_POST_OPT_NONE` POST to a single waiting task
    - `OS_POST_OPT_BROADCAST` POST to ALL tasks that are waiting on the queue
    - `OS_POST_OPT_FRONT` Simulates OSQPostFront()
- **Returns**：the same as OSQPost()

OSQPostOpt()

```
OS_Q       *pq;

if (pevent->OSEventGrp != 0x00)           // See if any task pending on queue
{
    if ((opt & OS_POST_OPT_BROADCAST) != 0x00)
          while (pevent->OSEventGrp != 0x00)
              OS_EventTaskRdy(pevent, msg, OS_STAT_Q);
    else
          OS_EventTaskRdy(pevent, msg, OS_STAT_Q);
    OS_Sched();
    return (OS_NO_ERR);
}
```

OSQPostOpt()

```
pq = (OS_Q *)pevent->OSEventPtr;          //Point to queue control block
if (pq->OSQEntries >= pq->OSQSize)        //Make sure queue is not full
    return (OS_Q_FULL);
if ((opt & OS_POST_OPT_FRONT) != 0x00)    //Do we post to the FRONT of the queue?
{
    if (pq->OSQOut == pq->OSQStart)       //Yes, Post as LIFO, Wrap OUT pointer if we
          pq->OSQOut = pq->OSQEnd;        //are at the 1st queue entry
    pq->OSQOut--;
    *pq->OSQOut = msg;                    //Insert message into queue
}
else                                      //No,  Post as FIFO
{
    *pq->OSQIn++ = msg;
    if (pq->OSQIn == pq->OSQEnd)          //Wrap IN ptr if we are at end of queue
        pq->OSQIn = pq->OSQStart;
}
pq->OSQEntries++;                         //Update the nbr of entries in the queue
return (OS_NO_ERR);
```

OSQAccept ()

- **Prototype**：`void *OSQAccept(OS_EVENT *pevent)`
- **Description**： This function checks the queue to see if a message is available.
- **Arguments**：
  - `pevent` is a pointer to the event control block.
- **Returns**：
  - `!= (void *)0` is the message in the queue if one is available.
  - `== (void *)0`
    - if the queue is empty
    - if 'pevent' is a NULL pointer
    - if you passed an invalid event type

OSQAccept()

```
void      *msg;
OS_Q      *pq;

pq = (OS_Q *)pevent->OSEventPtr;          //Point at queue control block
if (pq->OSQEntries > 0)                   //See if any messages in the queue
{
    msg = *pq->OSQOut++;                  //Yes, extract oldest message from the queue
    pq->OSQEntries--;                     //Update the number of entries in the queue
    if (pq->OSQOut == pq->OSQEnd)         //Wrap OUT pointer if we are at the end of the queue
          pq->OSQOut = pq->OSQStart;
}
else
    msg = (void *)0;                        /*Queue is empty
return (msg);                               /*Return message received (or NULL)
```

OSQQuery()

- **Prototype**：`INT8U OSQQuery(OS_EVENT *pevent, OS_Q_DATA *pdata)`
- **Description**：This function obtains information about a message queue.
- **Arguments**：
  - `pevent` is a pointer to the event control block.
  - `pdata` is a pointer to a structure that will contain information about the message queue.
- **Returns**：
  - `OS_NO_ERR`
  - `OS_ERR_EVENT_TYPE`
  - `OS_ERR_PEVENT_NULL`

OSQQuery()

```
typedef struct
{
  void    *OSMsg;              //Pointer to next message to be extracted from queue
  INT16U  OSNMsgs;             //Number of messages in message queue
  INT16U  OSQSize;             //Size of message queue
  INT8U   OSEventTbl[OS_EVENT_TBL_SIZE]; //List of tasks waiting for event to occur
  INT8U   OSEventGrp;          //Group corresponding to tasks waiting for event to occur
} OS_Q_DATA;
```

OSQQuery()

```
  OS_Q      *pq;
  INT8U     *psrc;
  INT8U     *pdest;
  pdata->OSEventGrp = pevent->OSEventGrp;          //Copy message queue wait list
  psrc              = &pevent->OSEventTbl[0];
  pdest             = &pdata->OSEventTbl[0];
#if OS_EVENT_TBL_SIZE > 0
    *pdest++          = *psrc++;
#endif
           :
           :
#if OS_EVENT_TBL_SIZE > 7
    *pdest            = *psrc;
#endif
  pq = (OS_Q *)pevent->OSEventPtr;
  if (pq->OSQEntries > 0)
      pdata->OSMsg = *pq->OSQOut;        //Get next message to return if available
  else
      pdata->OSMsg = (void *)0;
  pdata->OSNMsgs = pq->OSQEntries;
  pdata->OSQSize = pq->OSQSize;
  return (OS_NO_ERR);
```

Using a Message Queue as a Counting Semaphore

- Initializing queue with many non-NULL pointers as resources are available.
- Signal：OSQPost()
- Wait：OSQPend()

Using a Mailbox Instead of OSTimeDly()

- Example：
  - OSQPend(QTimeDly, TIMEOUT, &err);