

## MicroC-OS-II

(synchronization 1)

## Outline

- Task synchronization
  - Event control block
  - Mailbox
  - Message queue
  - Semaphore
  - Mutex semaphore
  - Event flag



## Event control block -overview

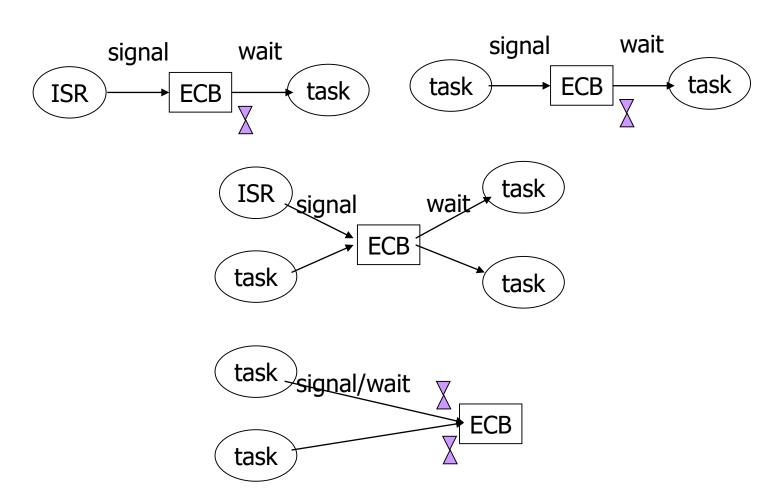
- Semaphore, message queue, mailbox
  - OSXXXCreate
  - OSXXXDel
  - OSXXXPend
    - Wait for XXX
  - OSXXXPost
    - Release XXX
  - OSXXXAccept
    - A non-blocking version of OSXXXPend
  - OSXXXQuery



- Semaphores, mutexs, message queues, mailboxes are managed by event control blocks
- For example, a semaphore is created
  - A event control block associated with the semaphore is also created



## In other words;





## **EVENT CONTROL BLOCK**

Task 1



Task 2

**WAITING** 







OS_EVENT_TYPE							
OSEVENTCnt							
OSEVENTPtr							
OSEVENTGrp							
7	6	5	4	3	2	1	0
•••							
63	62	61	60	59	58	57	56



Data structure (uCOS\_II.H)

```
typedef struct {
  void *OSEventPtr;
  INT16U OSEventCnt;
  INT8U OSEventType;
  INT8U OSEventGrp;
  INT8U OSEventTbl[OS_EVENT_TBL_SIZE];
} OS_EVENT;
```



## OSEventType

- ECB의 용도를 나타내는 필드
- OS\_EVENT\_TYPE\_SEM, OS\_EVENT\_TYPE\_MUTEX, OS\_EVENT\_TYPE\_MBOX, OS\_EVENT\_TYPE\_Q
- OSEventCnt
  - 세마포어
    - 세마포어 카운트 저장
  - 상호배제 세마포어
    - PIP 저장
- OSEventPtr
  - 메일박스나 메시지 큐일때만 사용
  - 해당 구조체를 가리키는 포인터
- OSEventGrp, OSEventTbl[]
  - OSRdyTbl[], OSRdyGrp과 비슷한 형태
  - 이벤트의 발생을 기다리는 태스크들의 리스트



**EVENT CONTROL BLOCK** 

Task 1

Task 4

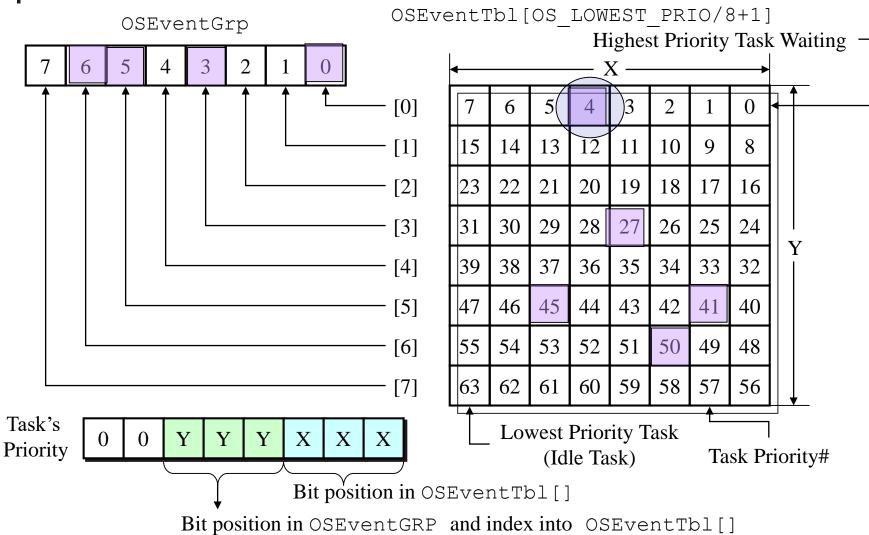
Task 27

Task 41

Task 45

Task 50







#### Insert a task to event wait list

#### Remove a task from event wait list

```
if ((pevent->OSEventTbl[prio >> 3] &= ~OSMapTbl[prio & 0x07]) == 0) {
    pevent->OSEventGrp &= ~OSMapTbl[prio >> 3];
}
```

### Finding the highest priority task from event wait list

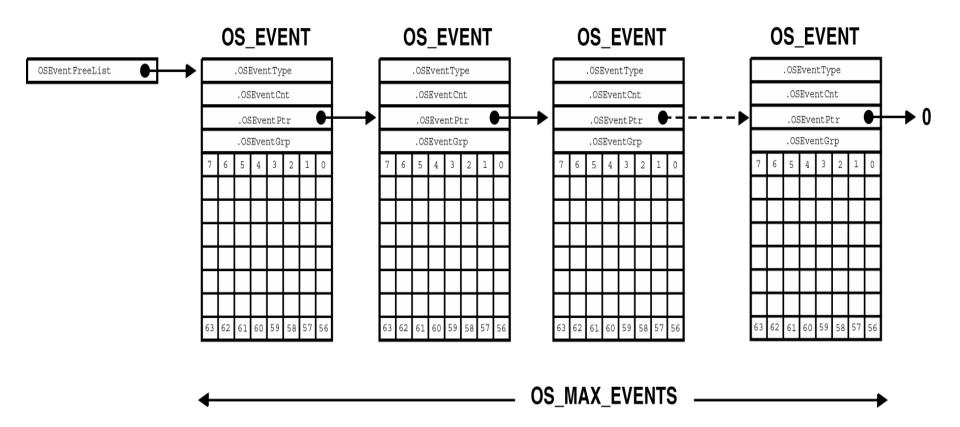
```
y = OSUnMapTbl[pevent->OSEventGrp];
x = OSUnMapTbl[pevent->OSEventTbl[y]];
prio = (y << 3) + x;</pre>
```



### Free event control block list

- 할당되는 ECB의 개수는 응용 프로그램에서 요구하는 세마포어, 메일박스, 메시지 큐에 따라 다름
  - ECB 의 최대 개수는 OS\_CFG.H 의 OS\_MAX\_EVENTS 에서 정의됨
  - OS\_INIT이 호출되면 모든 ECB 는 자유 이벤트 제어 블록 리스트에 연결
  - 세마포어나 메일박스 큐가 생성되면 자유 이벤트 제어 블록 리스트에서 삭제되고 초기화됨







## Event control block -operations

- ECB operations
  - 1. ECB 초기화
    - OS\_EventWaitListInit()
  - 2. 태스크를 준비 상태로 만들기
    - OS\_EventTaskRdy()
  - 3. 태스크를 어떤 이벤트에 대해 wait 상태로 만들 기
    - OS\_EventWait()
  - 4. 이벤트를 기다리다가 timeout이 발생하여, 태스 크를 준비 상태로 만들기
    - OS\_EventTO()



OSXXXCreate
OSXXXDel
OSXXXPend
OSXXXPost
OSXXXAccept
OSXXXQuery



OS\_EventWaitListInit()
OS\_EventTaskRdy()
OS\_EventWait()
OSEventTO()



- ECB 제어 블록 관련 기능
  - OSEventWaitListInit(OS\_EVENT \*pevent)
    - Is called by OSxxxCreate()
    - 세마포어, 메일박스, 메시지 큐 생성 시에 호출되는 함수
    - 단지 ECB를 기다리는 태스크의 리스트가 아직은 없다는 것을 표시 하는 것임
  - OSEventTaskRdy()
    - Is called by Post
    - ECB가 시그널을 받아서, ECB를 기다리던 태스크중 가장 우선순위 가 높은 태스크를 실행 가능 상태로 바꿀 때
  - OSEventTaskWait()
    - Is called by PEND
    - 태스크를 준비 리스트에서 삭제
    - ECB 의 대기 리스트에 태스크 삽입
  - OSEventTO()
    - Is called by PEND
    - 타임아웃으로 태스크 준비 상태 만들기
    - OSSemPend(), OSMboxPend(), OSQPend() 에서 호출



```
void OS_EventTaskRdy(OS_EVENT *pevent, void *msg, INT8U msk)
  y = OSUnMapTbl[pevent->OSEventGrp];
 bity = OSMapTbl[y];
 x = OSUnMapTbl[pevent->OSEventTbl[y];우선순위가 높은 태스크 찾기
  bitx = OSMapTbl[x];
 prio = (INT8U) ((y>>3) + x);
 if ((pevent->OSEventTbl[y] &= \simbitx) == 0){
   pevent->OSEventGrp &= ~bity;
                                     ECB 대기 리스트에서 삭제
  ptcb
              = OSTCBPrioTbl[prio];
  ptcb->OSTCBDly = 0;
  ptcb->OSTCBEventPtr = (OS EVENT *) 0;
                                            OS STAT SEM, OS STAT MUTEX
  ptcb->OSTCBStat &= ~msk;
                                          → OS_STAT_MBOX, OS_STAT_Q
  if (ptcb->OSTCBStat == OS_STAT_RDY) {
   OSRdyGrp |= bity;
                                           준비 리스트 삽입
   OSRdyGrp |= bitx;
```



```
void OS_EventTaskWait(OS_EVENT *pevent)
{
    OSTCBCur->OSTCBEventPtr = pevent;
    if((OSRdyTbl[OSTCBCur->OSTCBY] &= ~OSTCBCur->OSTCBBitX) == 0) {
        OSRdyGrp &= ~OSTCBCur->OSTCBBitY;
    }
    pevent->OSEventTbl[OSTCBCur->OSTCBY] |= OSTCBCur->OSTCBBitX;
    pevent->OSEventGrp |= OSTCBCur->OSTCBBitY;
}

ECB 대기리스트에 삽입
```

```
void OS_EventTO (OS_EVENT *pevent)
{
    if((pevent->OSEventTbl[OSTCBCur->OSTCBY] &= ~OSTCBCur->OSTCBBitX)==0) {
        pevent->OSEventGrp &= ~OSTCBCur->OSTCBBitY;
    }
    OSTCBCur->OSTCBStat = OS_STAT_RDY;
    OSTCBCur->OSTCBEventPtr = (OS_EVENT *) 0;
    않도록 세팅
}
```

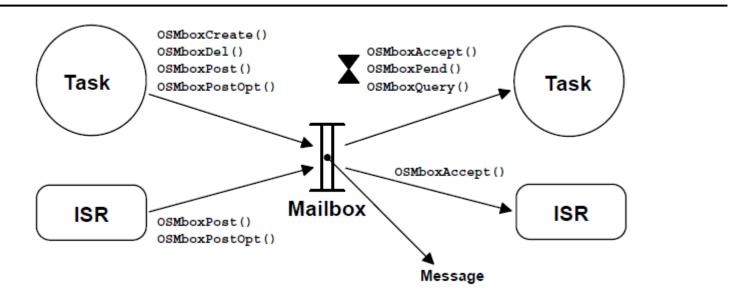
# Outline

- Task synchronization
  - Event control block
  - Mailbox
  - Message queue

## Message Mailbox

- Simply Mailbox
  - An object that allows a task or ISR to send a pointer-based variable to another task.

Figure 10.1 Relationships between tasks, ISRs, and a message mailbox.





## Functions on Mailbox (OS\_CFG.H)

- mailbox 생성
  - OS\_EVENT \*OSMboxCreate (void \*msg)
    - msg: 메시지에 대한 포인터
    - pevent->OSEventPtr = msg;
- mailbox 삭제
  - OS\_EVENT \*OSMboxDel (OS\_EVENT \*pevent, INT8U opt, INT8U \*err)
  - Mailbox를 삭제하려면 그 Mailbox를 액세스할 수 있는 모든 Task를 먼저 삭제해야 함



- mailbox 에서 메시지 기다리기
  - void \*OSMboxPend (OS\_EVENT \*pevent, INT16U timeout, INT8U \*err)
    - .OSEventPtr이 NULL 이 아닌 포인터값을 가지고 있을 때
    - OSMboxPend()가 호출되면, .OSEventPtr값을 msg에 저 장하고, 메일박스는 비운다
    - .OSEventPtr에는 NULL pointer값을 저장
    - 만약 .OSEventPtr이 NULL 이면 waiting상태가 되며,
    - 메시지가 메일박스에 들어왔을 때, 그리고
       OSMboxPend()를 호출한 이 태스크가 다시 HPT가 되었을 때 OSSched()는 이 자리로 복귀하며, message는 호출한 태스크에 넘겨짐



- mailbox 에 메시지 보내기
  - INT8U OSMboxPost (OS\_EVENT \*pevent, void \*msg)
    - 메시지 저장
- 기다리지 않고 mailbox 에서 메시지 읽기
  - void \*OSMboxAccept (OS\_EVENT \*pevent)
- mailbox 대한 정보 얻기
  - INT8U OSMboxQuery (OS\_EVENT \*pevent, OS\_MBOX\_DATA \*pdata)

```
typedef struct {
   void *OSMsg;
   INT8U OSEventTbl[OS_EVENT_TBL_SIZE];
   INT8U OSEventGrp;
} OS_MBOX_DATA;
```



```
void *OSMboxPend (OS EVENT *pevent, INT16U timeout, INT8U *err)
                                                   0: wait forever
  void *msq;
   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();
                                   메시지가 있을 경우
  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 (msq);
                              /* Return the message received (or NULL) */
OSTCBCur->OSTCBStat |= OS STAT MBOX; /* Message not available, task will pend */
OSTCBCur->OSTCBDly = timeout; /* Logd timeout on 50 00 000
OS EventTaskWait(pevent);
OS EXIT CRITICAL();
OS Sched();
                              /* Find next highest priority task ready to run */
OS ENTER CRITICAL();
msq = OSTCBCur->OSTCBMsq;
if (msg != (void *)0) { /* See if we were given the message */
   OSTCBCur->OSTCBMsq = (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 (msq);
                              /* Return the message received */
OS EXIT CRITICAL();
*err = OS TIMEOUT;
                             /* Indicate that a timeout occurred */
return ((void *)0);
                              /* Return a NULL message */
```



```
INT8U OSMboxPost (OS EVENT *pevent, void *msg)
#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) / 1
                           /* Ready highest priority task waiting on event */
      OS EXIT CRITICAL();
                           /* Find highest priority task ready to run */
      OS Sched();
      return (OS NO ERR);
```





### **EVENT CONTROL BLOCK**

Task 1



OSMboxPost



Task 2

<1>

OSMboxPend WAITING

<2>





TxMbox = OSMboxCreate((void \*)0);

char \*rxmsg;
 INT8U err;

rxmsg = (char \*)OSMboxPend(TxMbox, 0, &err);

char txmsg;
OSMboxPost(TxMbox, (void \*)&txmsg);





## OSMboxPostOpt()

- Mailbox에 대기 중인 모든 Task에게 메시지를 Broadcast할 수 있는 기능 추가
  - 전달인자
    - Pevent: 해당 Mailbox의 ECB에 대한 포인터
    - Timeout: 옵션 타임아웃 값(Clock Tick 단위)
    - Broadcast option : OS\_POST\_OPT\_BROADCAST



```
INT8U OSMboxPostOpt (OS EVENT *pevent , void *msg/ INT8U opt)
#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->0SEventGrp != 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, msq, 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= msq; /* Place message in mailbox */
OS EXIT CRITICAL();
return (OS NO ERR);
```



## OSMboxAccept()

- 호출 Task를 대기상태로 만들지 않고 메시지를 받을 수 있음
- 전달인자
  - ▶ Pevent: 해당 Mailbox의 ECB에 대한 포인터



```
void *OSMboxAccept (OS EVENT *pevent)
   void * msg;
   OS ENTER CRITICAL();
   if (pevent->OSEventType != OS_EVENT_TYPE_MBOX) {
      OS EXIT CRITICAL();
      return ((void *)0);
   msg = pevent->OSEventPtr;
   if (msg != (void *)0) {
      pevent->OSEventPtr = (void *)0;
   OS EXIT CRITICAL();
   return (msg);
```

# Outline

- Task synchronization
  - Event control block
  - Mailbox
  - Message queue

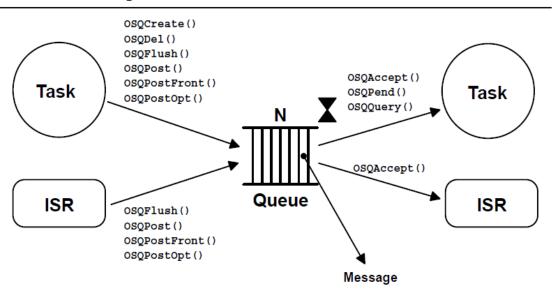


## Message Queue

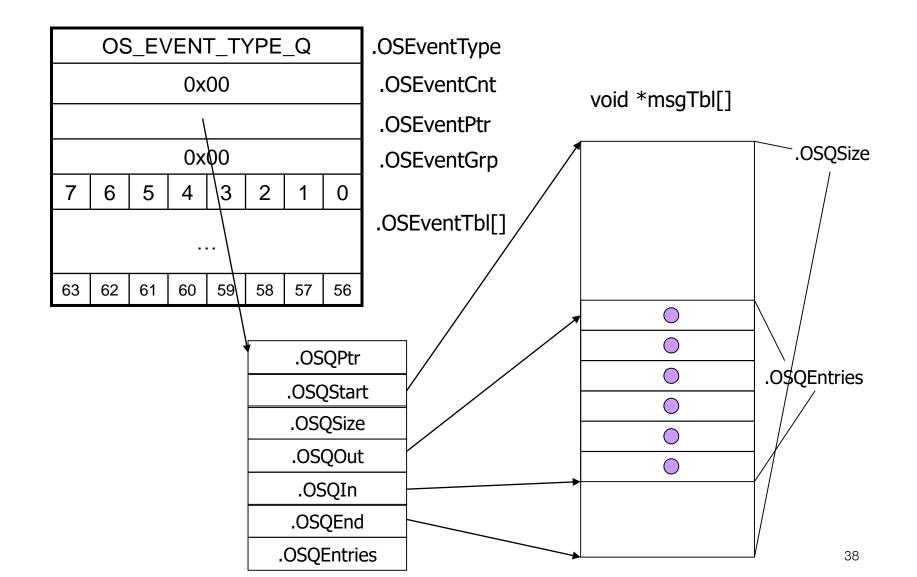
## Simply Queue

 An object that allows a task or ISR to send pointerbased variables to another task.

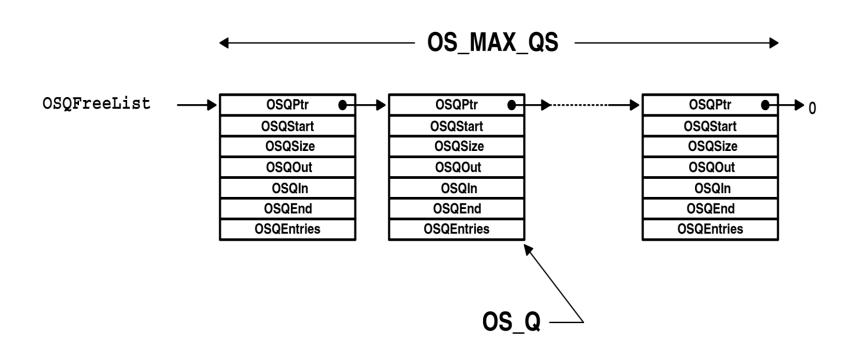
Figure 11.1 Relationships between tasks, ISRs, and a message queue.



# Message Queue









- Message queue 생성
  - OS\_EVENT \*OSQCreate (void \*\*start, INT16U size)
    - 메시지를 담을 포인터 배열이 미리 할당되어야 함
- Message queue에 메시지 오기를 기다림
  - void \*OSQPend (OS\_EVENT \*pevent, INT16U timeout, INT8U \*err)
    - 메시지가 큐에 도착하기를 기다리는 함수
      - 만약 큐 속에 저장되어 있던 메시지가 없다면, 새로운 메시 지가 도착하거나, timeout 시간이 지날 때까지 태스크를 sleep 시키고는 scheduler(OSSched())를 호출한다.
      - 다시 이 OSQPend()가 HPT가 되었을 때, 새로 들어온 메 시지가 있다면 메시지를 꺼낸다.



- Message queue에 배달된 모든 메시지 삭제
  - INT8U OSQFlush (OS\_EVENT \*pevent)
- Broadcasting
  - INT8U OSQPostOpt (OS\_EVENT \*pevent, void \*msg, INT8U opt)

## OS\_POST\_OPT\_BROADCAST

- Non-blocking message queue
  - void \*OSQAccept (OS\_EVENT \*pevent)



```
#define
          MSG_QUEUE_SIZE
                            20
void
          *MsgQueueTbl[20];
                                       /* Storage for messages */
MsgQueue = OSQCreate(&MsgQueueTbl[0], MSG_QUEUE_SIZE);
char *msg;
INT8U err;
msg = (char *)OSQPend(MsgQueue, 0, &err);
char msg[20];
strcpy(&msg[0], "Task 3");
OSQPost(MsgQueue, (void *)&msg[0]);
```