实时内核

任务之间的通信与同步

事件控制块ECB

- 所有的通信信号都被看成是事件(event)，μC/OS-II通过事件控制块(ECB)来管理每一个具体事件

ECB数据结构

```
typedef struct {
    void *OSEventPtr;  /*指向消息或消息队列的指针*/
    INT8U OSEventTbl[OS_EVENT_TBL_SIZE];//等待任务列表
    INT16U OSEventCnt; /*计数器（当事件是信号量时）*/
    INT8U OSEventType; /*事件类型：信号量、邮箱等*/
    INT8U OSEventGrp;  /*等待任务组*/
} OS_EVENT;
```

与TCB类似的结构，使用两个链表，空闲链表与使用链表

事件控制块ECB数据结构

7	6	5	4	3	2	1	0
15	14	13	12	11	10	9	8
23	22	21	20	19	18	17	16
31	30	29	28	27	26	25	24
39	38	37	36	35	34	33	32
47	46	45	44	43	42	41	40
55	54	53	52	51	50	49	48
63	62	61	60	59	58	57	56

OSEventTbl[8]

任务和ISR之间的通信方式

- 一个任务或ISR可以通过事件控制块ECB（信号量、邮箱或消息队列）向另外的任务发信号
- 一个任务还可以等待另一个任务或中断服务子程序给它发送信号，对于处于等待状态的任务，还可以给它指定一个最长等待时间
- 多个任务可以同时等待同一个事件的发生，当该事件发生后，在所有等待该事件的任务中，优先级最高的任务得到了该事件并进入就绪状态，准备执行

等待任务列表

- 每个正在等待某个事件的任务被加入到该事件的ECB的等待任务列表中，该列表包含两个变量OSEventGrp和OSEventTbl[ ]
- 在OSEventGrp中，任务按优先级分组，8个任务为一组，共8组，分别对应OSEventGrp 当中的8位
- 当某组中有任务处于等待该事件的状态时，对应的位就被置位，同时OSEventTbl[]中的相应位也被置位

OSEventGrp

7	6	5	4	3	2	1	0

OSEventTbl[OS_LOWEST_PRIO / 8+1]

最高优先级任务

X

Y

[0]	7	6	5	4	3	2	1	0
[1]	15	14	13	12	11	10	9	8
[2]	23	22	21	20	19	18	17	16
[3]	31	30	29	28	27	26	25	24
[4]	39	38	37	36	35	34	33	32
[5]	47	46	45	44	43	42	41	40
[6]	55	54	53	52	51	50	49	48
[7]	63	62	61	60	59	58	57	56

任务的优先级

0	0	Y	Y	Y	X	X	X

OSEventTbl[ ]中相应位的位置

OSEventGrp 中相应位的位置及
OSEventTbl[ ]中的数组下标

最低优先级任务(即空闲任务，不可能处于等待状态)

正在等待该事件的任务的优先级

使任务进入/脱离等待状态

- 将一个任务插入到事件的等待任务列表中

```
pevent->OSEventGrp           |= OSMapTbl[prio >> 3];
pevent->OSEventTbl[prio >> 3] |= OSMapTbl[prio & 0x07];
```

- 从等待任务列表中删除一个任务

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

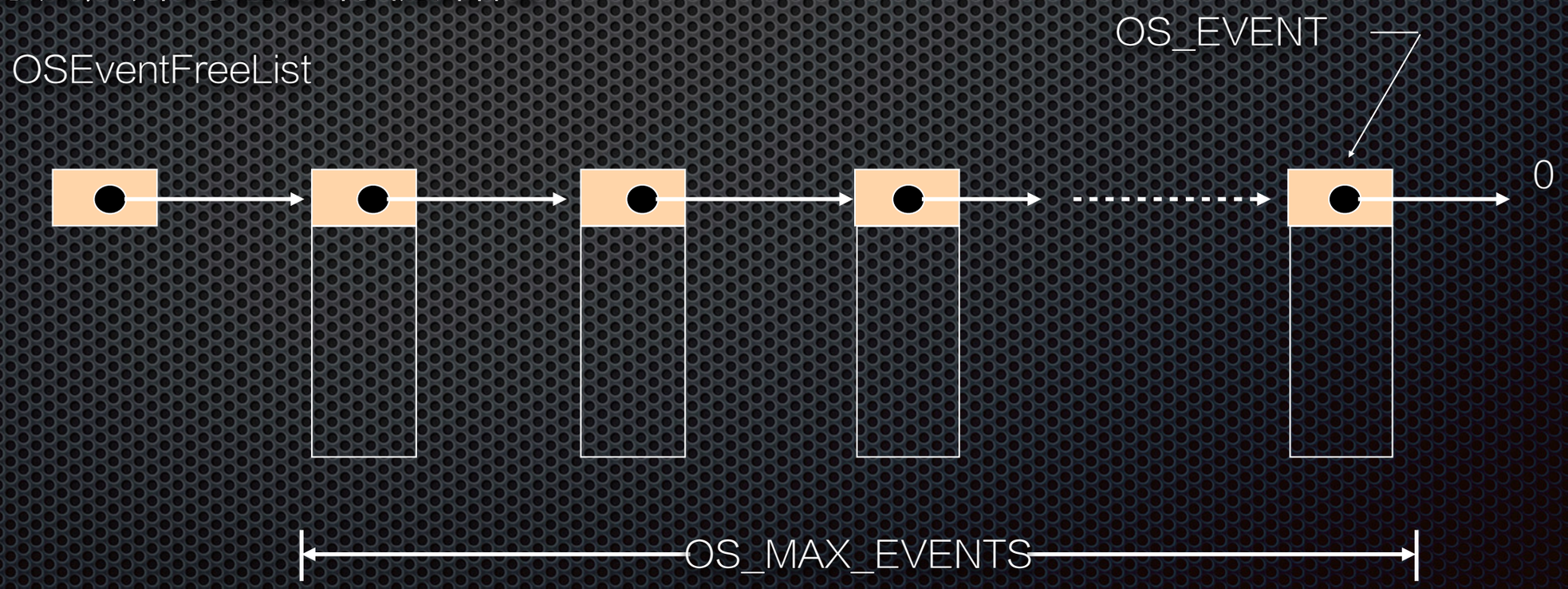
在等待事件的任务列表中查找优先级最高的任务

- 在等待任务列表中查找最高优先级的任务

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

空闲ECB的管理

- ECB的总数由用户所需要的信号量、邮箱和消息队列的总数决定，由OS_CFG.H中的#define OS_MAX_EVENTS定义
- 在调用OSInit()初始化系统时，所有的ECB被链接成一个单向链表——空闲事件控制块链表
- 每当建立一个信号量、邮箱或消息队列时，就从该链表中取出一个空闲事件控制块，并对它进行初始化



任务间通信

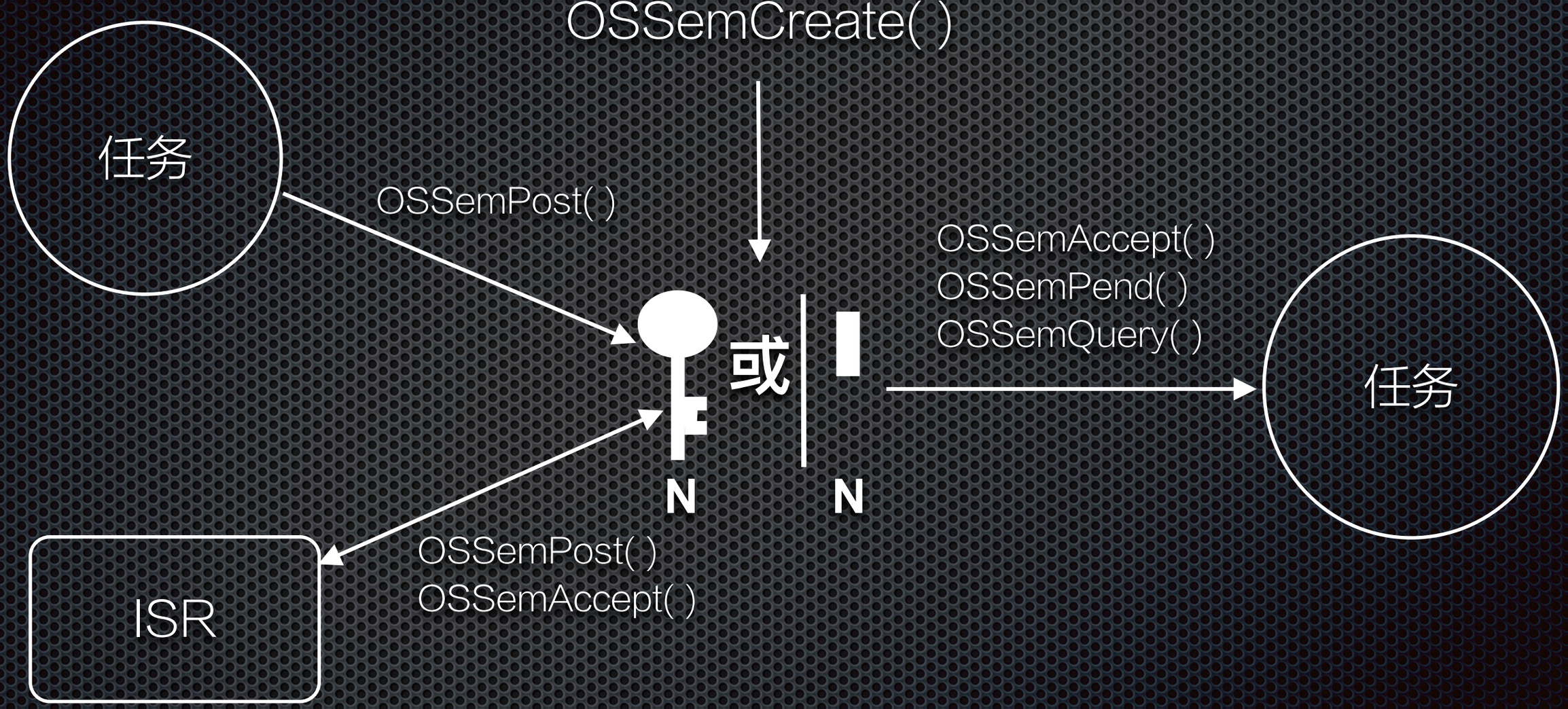
- 低级通信
  - 只能传递状态和整数值等控制信息，传送的信息量小
  - 例如：信号量
- 高级通信
  - 能够传送任意数量的数据
  - 例如：共享内存、邮箱、消息队列

信号量

- 信号量在多任务系统中的功能
  - 实现对共享资源的互斥访问（包括单个共享资源或多个相同的资源）
  - 实现任务之间的行为同步
- 必须在OS_CFG.H中将OS_SEM_EN开关常量置为1，这样μC/OS才能支持信号量

- uC/OS中信号量由两部分组成：
  - 信号量的计数值（16位无符号整数）
  - 等待该信号量的任务所组成的等待任务表
- 信号量系统服务
  - OSSemCreate()
  - OSSemPend(), OSSemPost()
  - OSSemAccept(), OSSemQuery()

任务、ISR和信号量的关系



共享内存

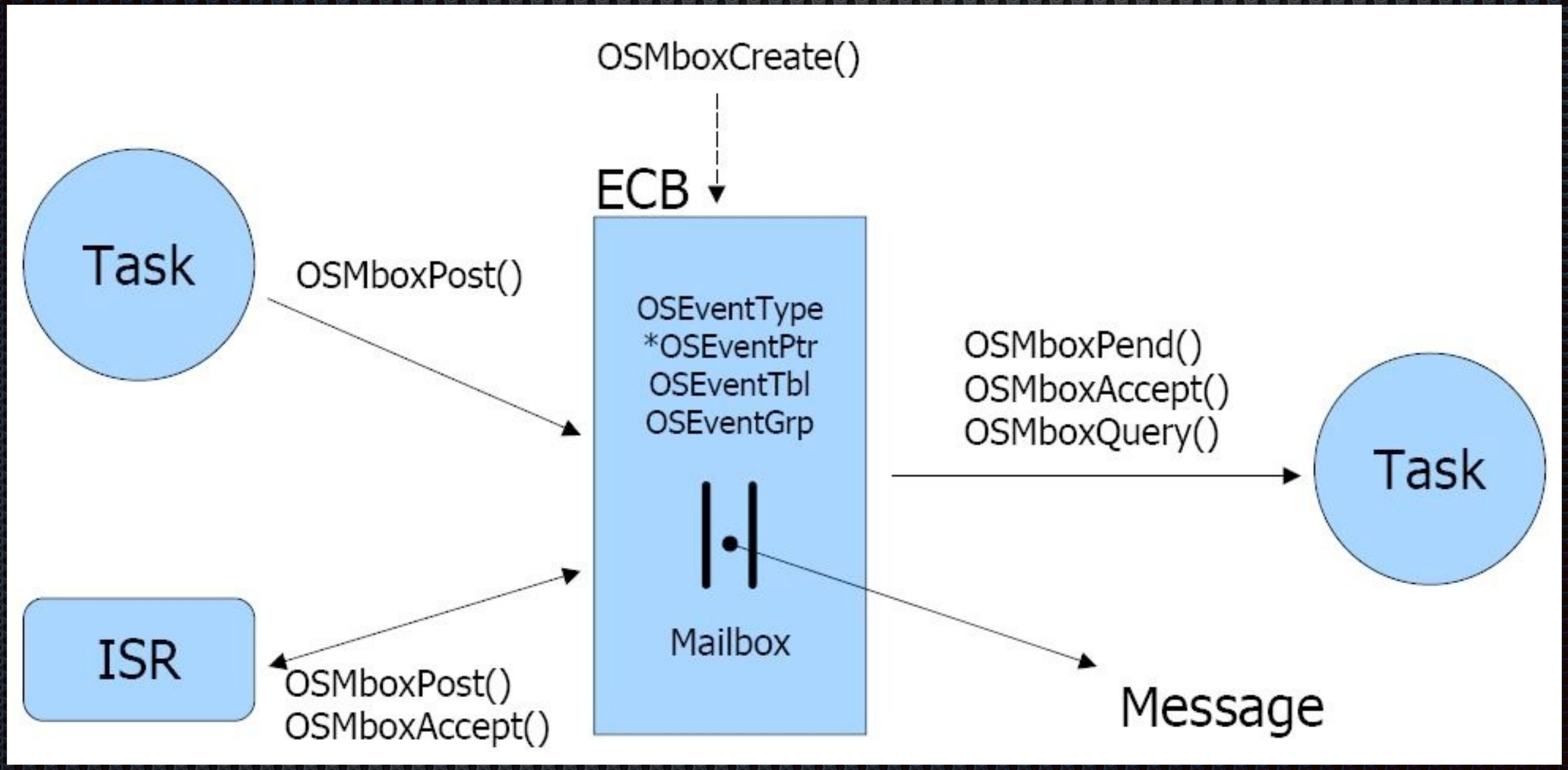
- 在μC/OS-II中如何实现共享内存?
  - 内存地址空间只有一个，为所有的任务所共享
  - 为了避免竞争状态，需要使用信号量来实现互斥访问

消息邮箱

- 邮箱（MailBox）：一个任务或ISR可以通过邮箱向另一个任务发送一个指针型的变量，该指针指向一个包含了特定“消息”（message）的数据结构；
- 必须在OS_CFG.H中将OS_MBOX_EN开关常量置为1，这样μC/OS才能支持邮箱。

- 一个邮箱可能处于两种状态：
  - 满的状态：邮箱包含一个非空指针型变量
  - 空的状态：邮箱的内容为空指针NULL
- 邮箱的系统服务
  - OSMboxCreate()
  - OSMboxPost()
  - OSMboxPend()
  - OSMboxAccept()
  - OSMboxQuery()

任务、ISR和消息邮箱的关系

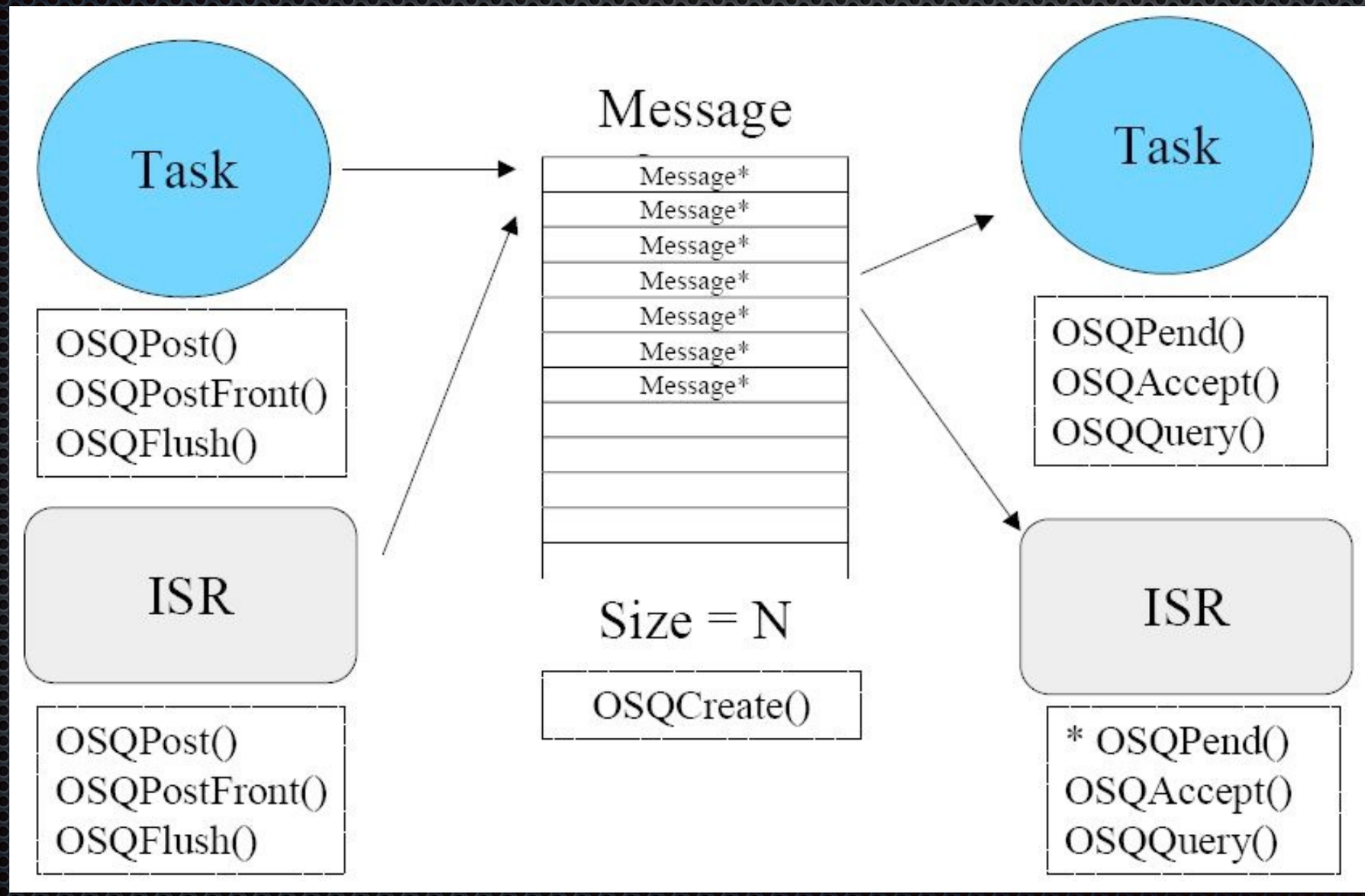


消息队列

- 消息队列（Message Queue）：消息队列可以使一个任务或ISR向另一个任务发送多个以指针方式定义的变量
- 为了使μC/OS能够支持消息队列，必须在OS_CFG.H中将OS_Q_EN开关常量置为1，并且通过常量OS_MAX_QS来决定系统支持的最多消息队列数

- 一个消息队列可以容纳多个不同的消息，因此可把它看作是由多个邮箱组成的数组，只是它们共用一个等待任务列表
- 消息队列的系统服务
  - OSQCreate()
  - OSQPend()、OSQAccept()
  - OSQPost()、OSQPostFront()
  - OSQFlush()
  - OSQQuery()

消息队列的体系结构



回忆一下ECB数据结构

- 在实现消息队列时，哪些字段可以用?

ECB数据结构

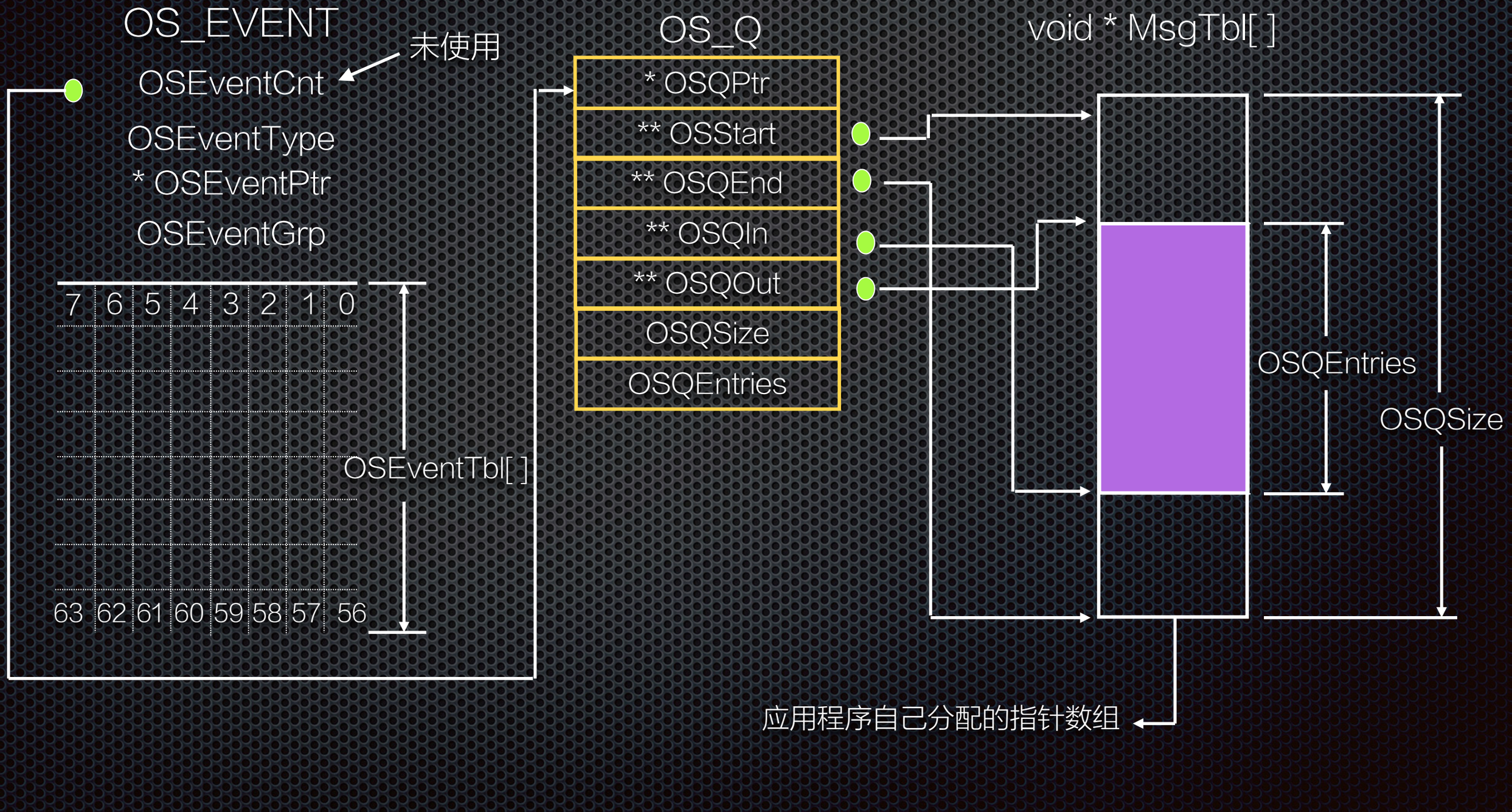
```
typedef struct {
    void *OSEventPtr;  /*指向消息或消息队列的指针*/
    INT8U OSEventTbl[OS_EVENT_TBL_SIZE];//等待任务列表
    INT16U OSEventCnt; /*计数器（当事件是信号量时）*/
    INT8U OSEventType; /*事件类型：信号量、邮箱等*/
    INT8U OSEventGrp;  /*等待任务组*/
} OS_EVENT;
```

队列控制块

- 队列控制块数据结构

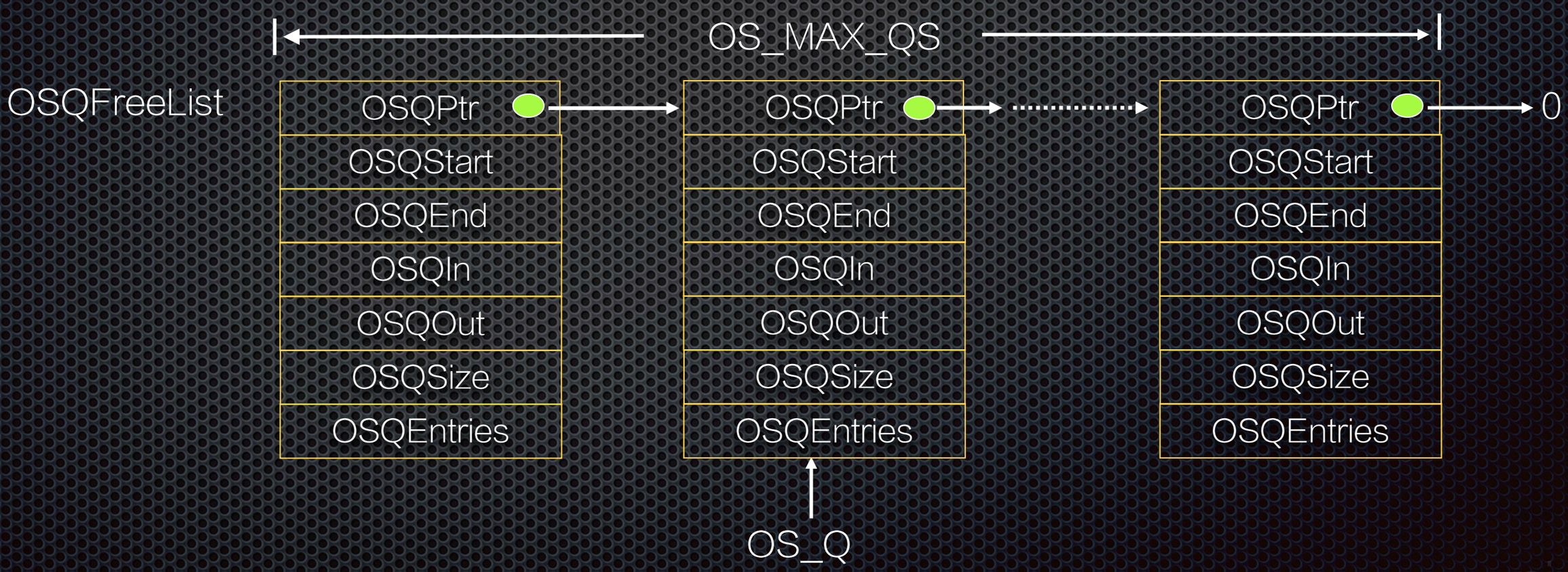
```
typedef struct os_q {
    struct os_q *OSQPtr;//空闲队列控制块指针
    void    **OSQStart; //指向消息队列的起始地址
    void    **OSQEnd;   //指向消息队列的结束地址
    void    **OSQIn; //指向消息队列中下一个插入消息的位置
    void    **OSQOut;//指向消息队列中下一个取出消息的位置
    INT16U  OSQSize;    //消息队列中总的单元数
    INT16U  OSQEntries; //消息队列中当前的消息数量
} OS_EVENT;
```

消息队列的数据结构

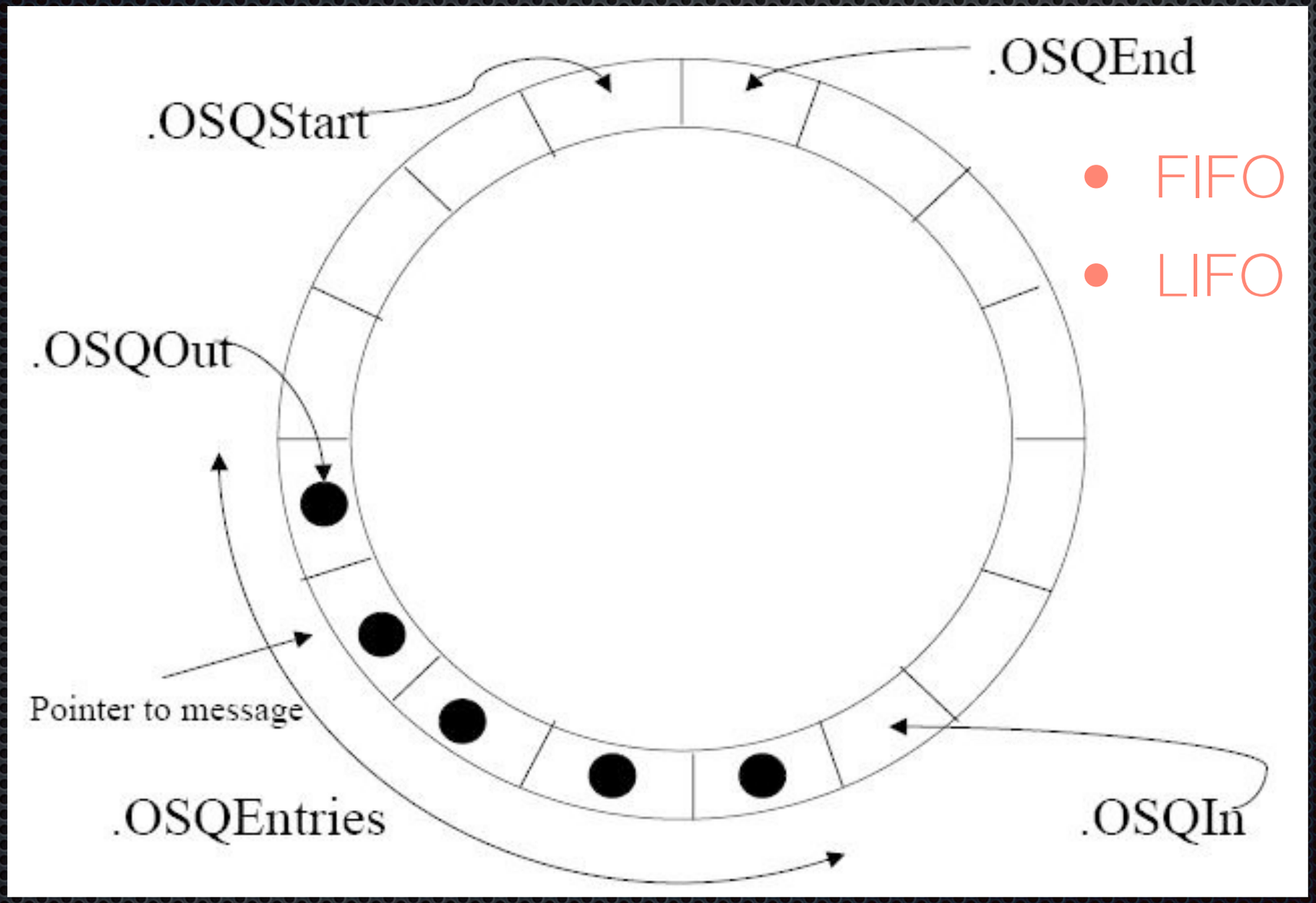

OS_EVENT
未使用
OSEventCnt
OSEventType
* OSEventPtr
OSEventGrp
7 6 5 4 3 2 1 0
63 62 61 60 59 58 57 56
OSEventTbl[ ]
OS_Q
* OSQPtr
** OSStart
** OSQEnd
** OSQIn
** OSQOut
OSQSize
OSQEntries
void * MsgTbl[ ]
OSQEntries
OSQSize
应用程序自己分配的指针数组

空闲队列控制块的管理

- 每一个消息队列都要用到一个队列控制块
- 在μC/OS中，队列控制块的总数由OS_CFG.H中的常量OS_MAX_QS定义
- 在系统初始化时，所有的队列控制块被链接成一个单向链表——空闲队列控制块链表OSQFreeList



消息缓冲区



- FIFO
- LIFO

队列控制块与事件控制块