Chapter 6

Event Control Blocks

Speaker :

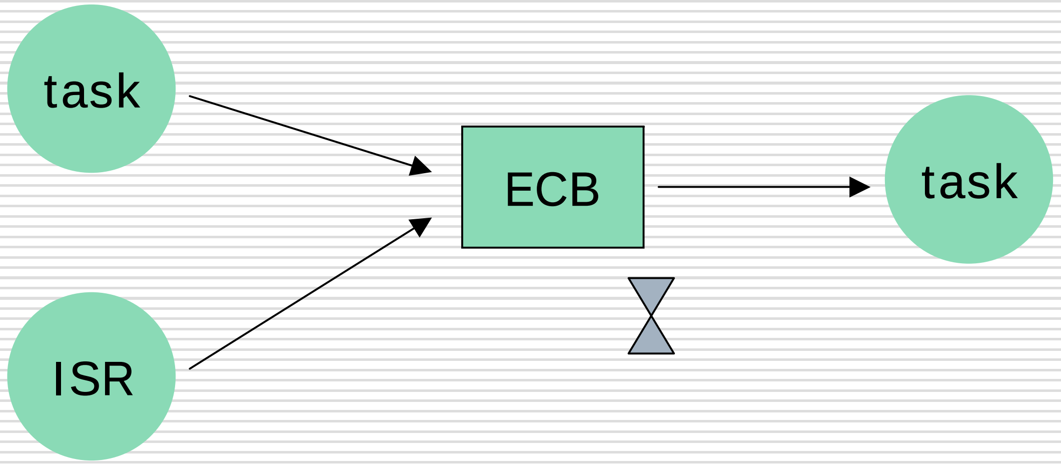
Shing-Guo Chang

Outline

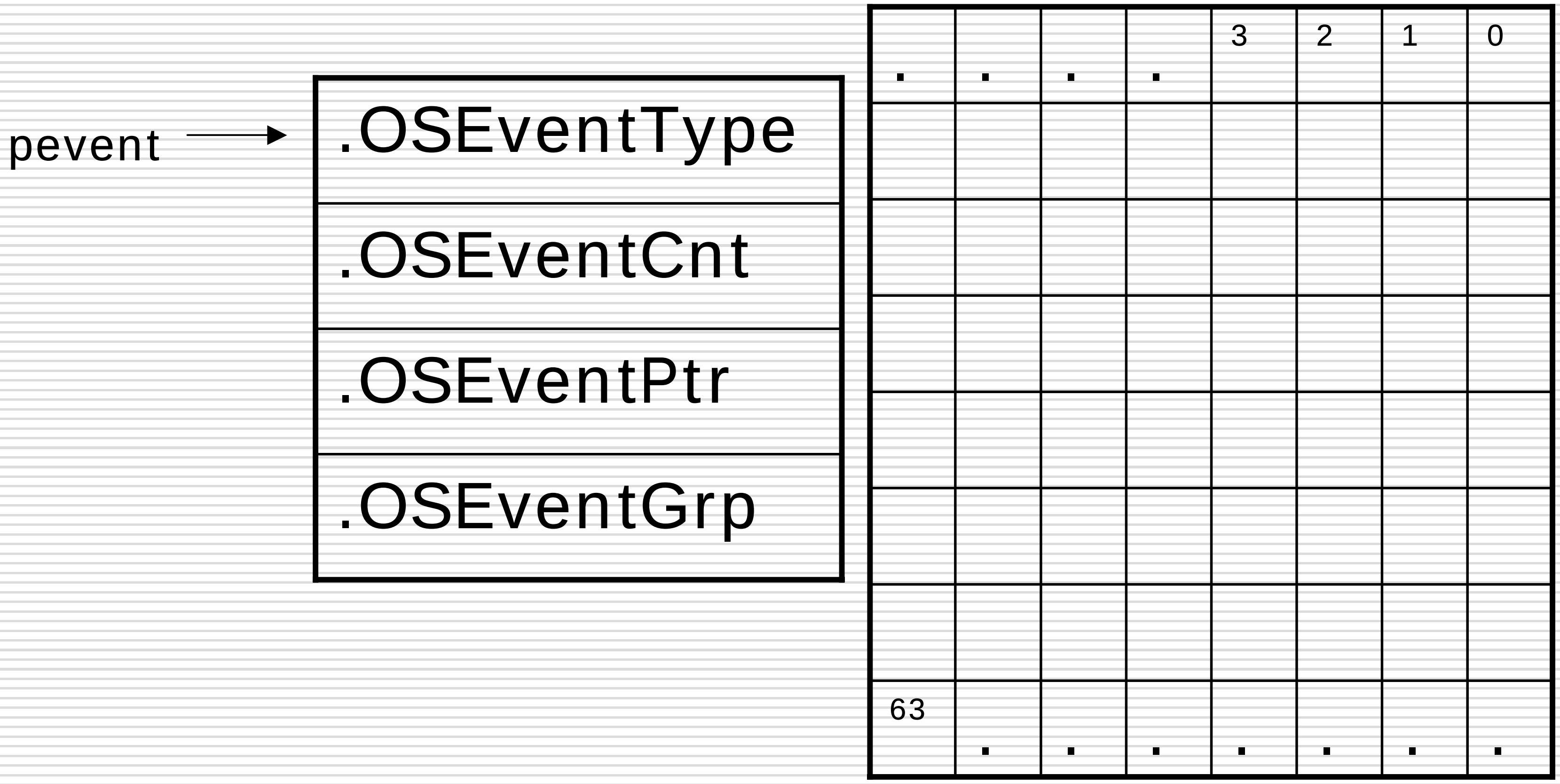
- Introduction
- Related Work
- Conclusion

Introduction(1/4)

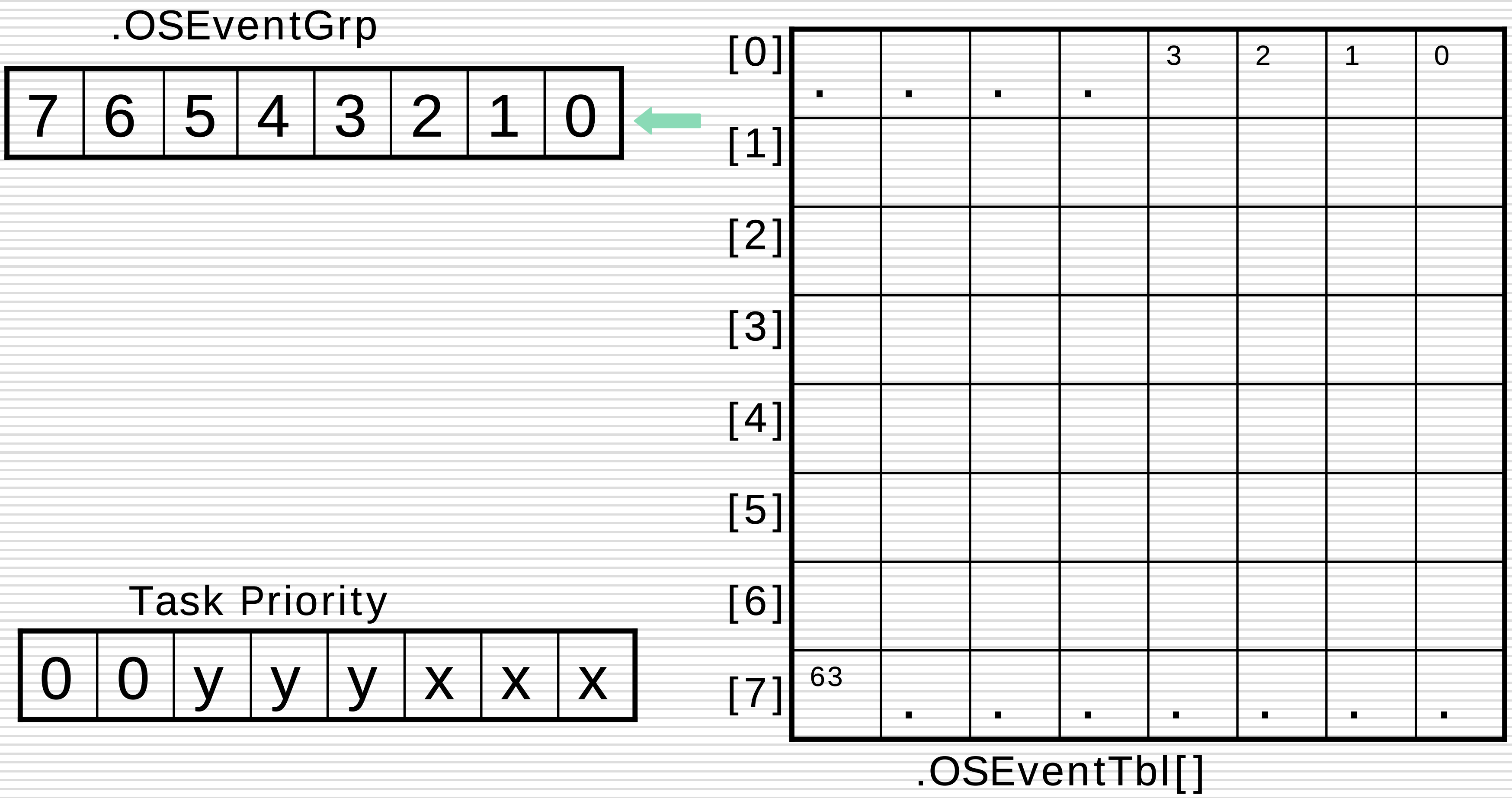
- ECB



ECB(cont.)


pevent
.OSEventType
.OSEventCnt
.OSEventPtr
.OSEventGrp
3
2
1
0
63

ECB(cont.)


.OSEventGrp
7 6 5 4 3 2 1 0
Task Priority
0 0 y y y x x x
[0]
[1]
[2]
[3]
[4]
[5]
[6]
[7]
3
2
1
0
63
.OSEventTbl[]

Introduction(2/4)

- Placing a task in the wait list

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

Introduction(3/4)

- Removeing a task from a wait list

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

Introduction(4/4)

- Find the highest priority task

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

Example

- .OSEventGrp contains 11001000(binary) and .OSEventTbl[3] contains 00010000(binary) , what is the waiting task priority ?

Some OSUnMapTb[]

```
4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0    /* 0x10 to 0x1F */
6, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0    /* 0xC0 to 0xCF */
```

Answer : prio=28

Related Work(1/4)

- Initializing an ECB

```
void OS_EventWaitListInit(OS_Event *pevent){
          INT8U *ptbl;
          pevent->OSEventGrp =0x00;
          ptbl                       = &pevent-> OSEventTbl[0];
#if OS_EVENT_TBL_SIZE >0
          *ptbl++          =0x00;
#if OS_EVENT_TBL_SIZE >1
          *ptbl++          =0x00;
#if OS_EVENT_TBL_SIZE >2
          *ptbl++          =0x00;
#if OS_EVENT_TBL_SIZE >3
          *ptbl++          =0x00;
```

Initializing an ECB(cont.)

```
#if OS_EVENT_TBL_SIZE >4
        *ptbl++         =0x00;
#if OS_EVENT_TBL_SIZE >5
        *ptbl++         =0x00;
#if OS_EVENT_TBL_SIZE >6
        *ptbl++         =0x00;
#if OS_EVENT_TBL_SIZE >7
        *ptbl    =0x00;
#endif
}
```

Related Work(2/4)

- Making a task ready

```
INT8U OS_EventTaskRdy(OS_EVENT *pevent, void *msg,
INT8U msk)
{
        OS_TCB *ptcb;
        INT8U x;
        INT8U y;
        INT8U bitx;
        INT8U bity;
        INT8U prio;

        y=OSUnMapTbl[pevent->OSEventGrp];
        bity=OSMaptbl[y];
        x=OSUnMapTbl[pevent->OSEventTbl[y]];
        bitx=OSMapTbl[x];
```

Making a task ready(cont.)

```
prio=(INT8U)((y<<3) + x);
if((pevent->OSEventTbl[y] &= ~bitx) = 0x00){
        pevent->OSEventGrp &= ~bity;
}
Ptcb =OSTCBPrioTbl[prio];
Ptcb->OSTCBEventPtr = (OS_EVENT *)0;
#if ((OS_Q_EN>0)&&(OS_MAX_OS>0)) || (OS_MBOX_EN >0)
        ptcb->OSTCBMsg = msg;
#else
        msg = msg;
#endif
        ptcb->OSTCBStat &= ~msk;
        if(ptcb->OSTCBStat = OS_STAT_RDY){
                OSRdyGrp    |=bity;
                OSRdyTbl[y]  |=bitx;
        }
        return(prio);
}
```

Related Work(3/4)

- Making a waiting task

```
void OS_EventTaskWait(OS_EVENT *pevent){
          OSTCBCur->OSTCBEventPtr = pevent ;
          if((OSRdyTbl[OSTCBCur->OSTCBY] &= ~OSTCBCur->
OSTCBBitx) = 0x00 ){
                    OSRdyGrp &= ~OSTCBBity;
          }
          pevent->OSEventTbl[OSTCBY] |= OSTCBCur->OSTCBBitX;
          pevent->OSEventGrp |=OSTCBCur->OSTCBBitY;
}
```

Related Work(4/4)

- Making a task ready because of time out

```
void OS_EventTO(OS_EVENT *pevent){
        if((pevent->OSEventTbl[OSTCBCur->OSTCBY] &=
~OSTCBCur->OSTCBBitx) = 0x00 ){
                pevent->OSEventGrp &= ~OSTCBCur->OSTCBBity;
        }
        OSTCBCur->OSTCBStat = OS_STAT_RDY;
        OSTCBCur->OSTCBEventPtr = (OS_EVENT *)0;
}
```

Conclusion

Thank you