A collection of military medals and a compass are arranged on a wooden surface. The medals include a red ribbon medal with a star, a blue ribbon medal with a star, and a silver star medal with a central emblem. A pair of glasses and a compass are also visible.

MicroC/OS-II

Kernel Structure-II

Santosh Sam Koshy
Member Technical Staff
C-DAC, Hyderabad

Task Control Blocks

- When a task is created it is assigned a Task Control Block **OS_TCB**
- **OS_TCB** is a Data Structure that is used by UCOS-II to maintain the state of a task when it is preempted
- All **OS_TCBs** reside in RAM


```

typedef struct os_tcb {
    OS_STK      *OSTCBStkPtr;                /* Pointer to current top of stack */

#ifdef OS_TASK_CREATE_EXT_EN
    void      *OSTCBExtPtr;                /* Pointer to user definable data for TCB extension */
    OS_STK      *OSTCBStkBottom;            /* Pointer to bottom of stack */
    INT32U      OSTCBStkSize;                /* Size of task stack (in bytes) */
    INT16U      OSTCBOpt;                    /* Task options as passed by OSTaskCreateExt() */
    INT16U      OSTCBId;                    /* Task ID (0..65535) */
#endif

    struct os_tcb *OSTCBNext;                /* Pointer to next TCB in the TCB list */
    struct os_tcb *OSTCBPrev;                /* Pointer to previous TCB in the TCB list */

#ifdef (OS_Q_EN && (OS_MAX_QS >= 2)) || OS_MBOX_EN || OS_SEM_EN
    OS_EVENT      *OSTCBEventPtr;            /* Pointer to event control block */
#endif

#ifdef (OS_Q_EN && (OS_MAX_QS >= 2)) || OS_MBOX_EN
    void      *OSTCBMsg;                    /* Message received from OSMboxPost() or OSQPost() */
#endif

    INT16U      OSTCBDly;                    /* Nbr ticks to delay task or, timeout waiting for event */
    INT8U      OSTCBStat;                    /* Task status */
    INT8U      OSTCBPrio;                    /* Task priority (0 == highest, 63 == lowest) */

    INT8U      OSTCBX;                      /* Bit position in group corresponding to task priority (0..7) */
    INT8U      OSTCBY;                      /* Index into ready table corresponding to task priority */
    INT8U      OSTCBBitX;                    /* Bit mask to access bit position in ready table */
    INT8U      OSTCBBitY;                    /* Bit mask to access bit position in ready group */


#ifdef OS_TASK_DEL_EN
    BOOLEAN      OSTCBDelReq;                /* Indicates whether a task needs to delete itself */
#endif
} OS_TCB;

```

OSTCBStkPtr

- Contains pointer to the current top-of-stack for the task
- Each task has its own stack & can be of any size
- OSTCBStkPtr is the only field in the OS_TCB that is accessed from assembly language code(from context switching code)

*OS_STK *OSTCBStkPtr;*



```
#if OS_TASK_CREATE_EXT_EN  
    void          *OSTCBExtPtr;  
    OS_STK        *OSTCBStkBottom;  
    INT32U        OSTCBStkSize;  
    INT16U        OSTCBOpt;  
    INT16U        OSTCBIId;  
#endif
```


OSTCBExtPtr

- Pointer to a user definable TCB extension
- This allows user to extend the TCB without having to change the UCOS-II source code
- It is used only by OSTaskCreateExt() so we need to set OS_TASK_CREATE_EXT_EN to 1

OSTCBStkBottom

- Pointer to the bottom of the tasks Stack
- If stack grows from high to low then points to lowest valid memory location for stack
- If stack grows from low to high then points to highest valid memory location for stack
- This allows you to determine free stack space for each task `OSTaskStkChk()`
- Set `OS_TASK_CREATE_EXT_EN` to 1

OSTCBStkSize


- Holds the size of the stack in number of elements instead of bytes
- This means that if the size is 1000 and each entry is 32-bits, the size of memory is 4000 Bytes

OSTCBOpt

- Holds options that can be passed to OSTaskCreateExt()
- Currently only three options defined:
 - OS_TASK_OPT_STK_CHK
 - OS_TASK_OPT_STK_CLR
 - OS_TASK_OPT_SAVE_FP

OSTCBIId

- Holds the Identifier for the task
- Has only been included for future expansion



*struct os_tcb *OSTCBNext;*
*struct os_tcb *OSTCBPrev;*

OSTCBNext & OSTCBPrev

- Used to doubly link OS_TCBS
- This chain is used by OSTimeTick() to update OSTCBDly for each task

OSTCBEventPtr

- Pointer to event control block

```
#if (OS_Q_EN && (OS_MAX_QS >= 2)) ||
```

```
OS_MBOX_EN || OS_SEM_EN  
OS_EVENT    *OSTCBEventPtr;
```

```
#endif
```

OSTCBMsg

- Pointer to the message that is sent to a task

```
#if (OS_Q_EN && (OS_MAX_QS >= 2)) ||  
    OS_MBOX_EN  
    void      *OSTCBMsg;  
#endif
```


OSTCBDly

- Used when task needs to be delayed for a certain number of clock ticks OR
- Needs to pend for a event to occur with a timeout
- (when 0 the task is not delayed or has no timeout while waiting for a event)



<i>INT16U</i>	<i>OSTCBDly;</i>
<i>INT8U</i>	<i>OSTCBStat;</i>
<i>INT8U</i>	<i>OSTCBPrio;</i>
<i>INT8U</i>	<i>OSTCBX;</i>
<i>INT8U</i>	<i>OSTCBY;</i>
<i>INT8U</i>	<i>OSTCBBitX;</i>
<i>INT8U</i>	<i>OSTCBBitY;</i>

OSTCBStat

- Contains the State of the task
- 0 => Ready to run etc
- Refer `ucos_ii.h`

OSTCBPrio

- Contains the priority of the task

OSTCBX, OSTCBY, OSTCBBitX, OSTCBBitY

- Used for speeding up the process of making a task ready to run or to make the task wait for a event
- Values for these variables are calculated when the task is created

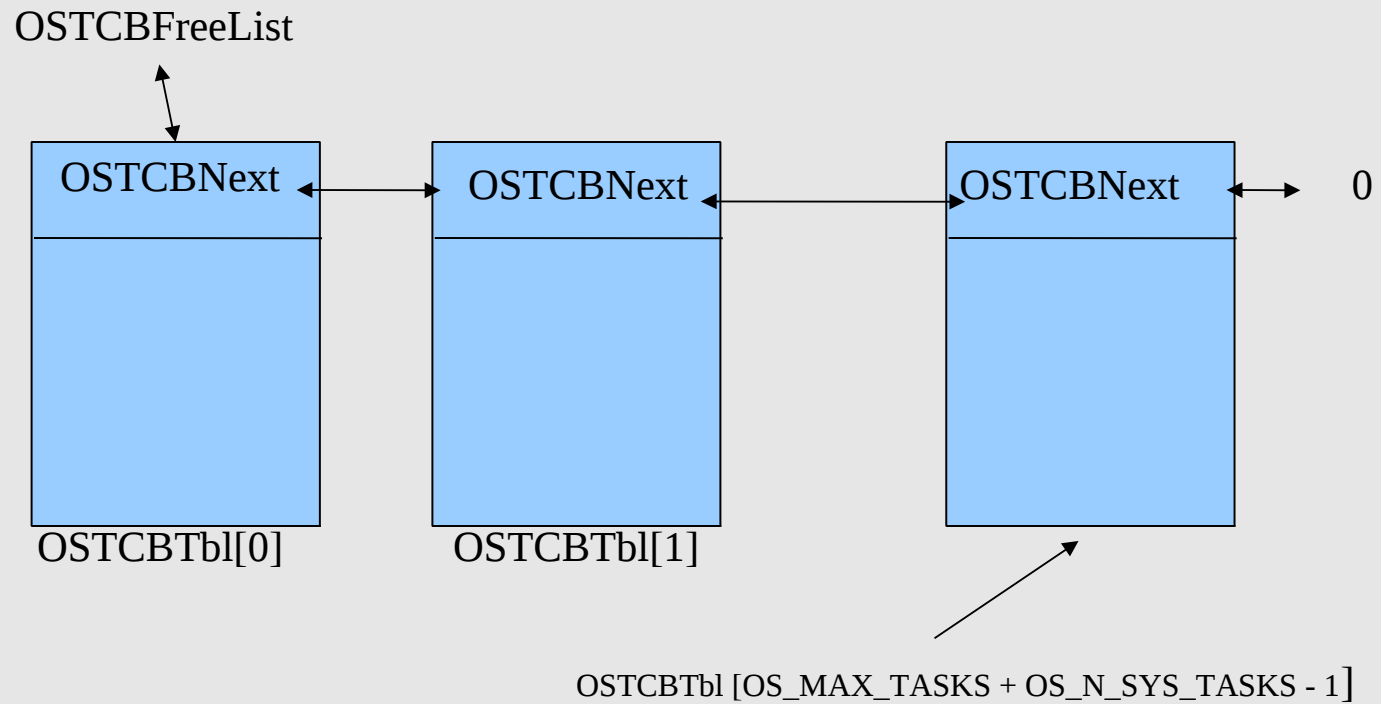
Calculations

OSTCBY = *priority* >> 3;
OSTCBBitY = *OSMapTbl*[*priority* >> 3];
OSTCBX = *priority* & 0x07;
OSTCBBitX = *OSMapTbl*[*priority* & 0x07];

Tasks and TCB Management

- The maximum number of tasks required by the application should be maintained in `OS_MAX_TASKS` and is found in `os_cfg.h`
- You can decrease the amount of memory maintained for the `OS_TCBs` by choosing appropriate number of tasks as reqd by the application
- An `OS_TCB` is initialized by a function `OS_TCBInit()` when a task is created using `OSTaskCreate()` or `OSTaskCreateExt()`

OSTCBFreeList



Ready List

- A task owns a priority level between 0 and OS_LOWEST_PRIO.
- Each task that is ready to run is placed in Ready List.

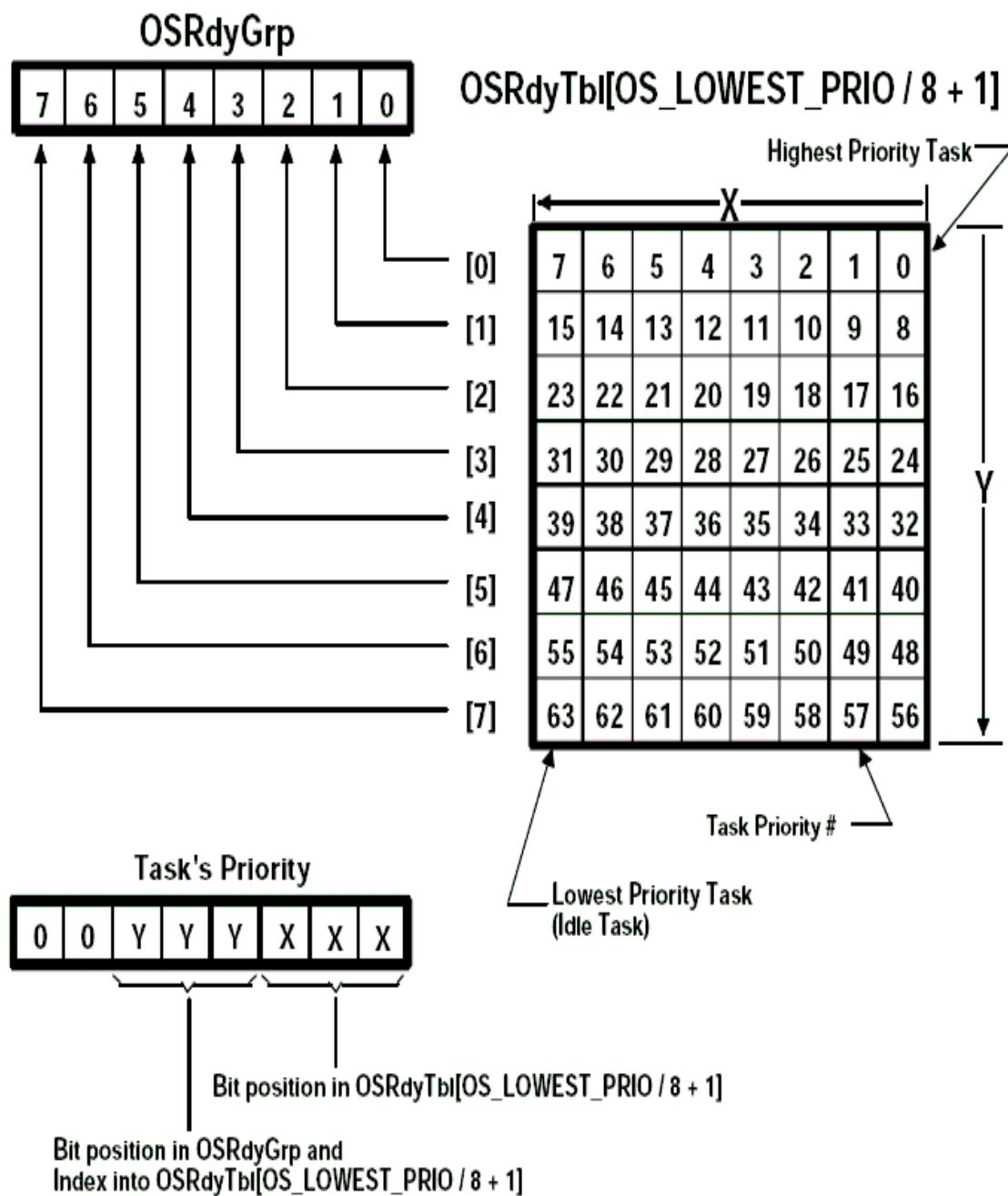
Components

● OSRdyGrp

- Task priorities are grouped (8 tasks per group).
- Each bit indicates a task of any group is ready to run.

● OSRdyTbl[]

- Size of OSRdyTbl[] is $OS_LOWEST_PRIO/8+1$.
- Task's priority determines the bit location in table.



Ready List

● OSMaPtbl[]

OSMaPtbl[0] == 000000001b

OSMaPtbl[1] == 00000010b

OSMaPtbl[2] == 00000100b

OSMaPtbl[3] == 00001000b

OSMaPtbl[4] == 00010000b

OSMaPtbl[5] == 00100000b

OSMaPtbl[6] == 01000000b

OSMaPtbl[7] == 10000000b

Ready List

- Make a task ready-to-run

```
OSRdyGrp          |= OSMapTbl[prio >> 3];  
OSRdyTbl[prio>>3] |= OSMapTbl[prio & 0x07];
```

- Remove a task from the ready list

```
if((OSRdyTbl[prio>>3]&=~OSMapTbl[prio&0x07  
    ]) == 0)  
    OSRdyGrp &= ~OSMapTbl[prio >> 3];
```


Ready List

- Find highest priority task ready-to-run

$y = \text{OSUnMapTbl}[\text{OSRdyGrp}];$

$x = \text{OSUnMapTbl}[\text{OSRdyTbl}[y]];$

$\text{prio} = (y \ll 3) + x;$

- OSUnMapTbl[]** priority resolution table

{ 0, 0, 1, 0, 2,
0, 1, 0, 3, 0,
1, 0, 2, 0, 1,
0, 4, 0, 1, 0,
2, 0, 1, 0, 3,
0, 1, 0, 2, 0,
... }



The End