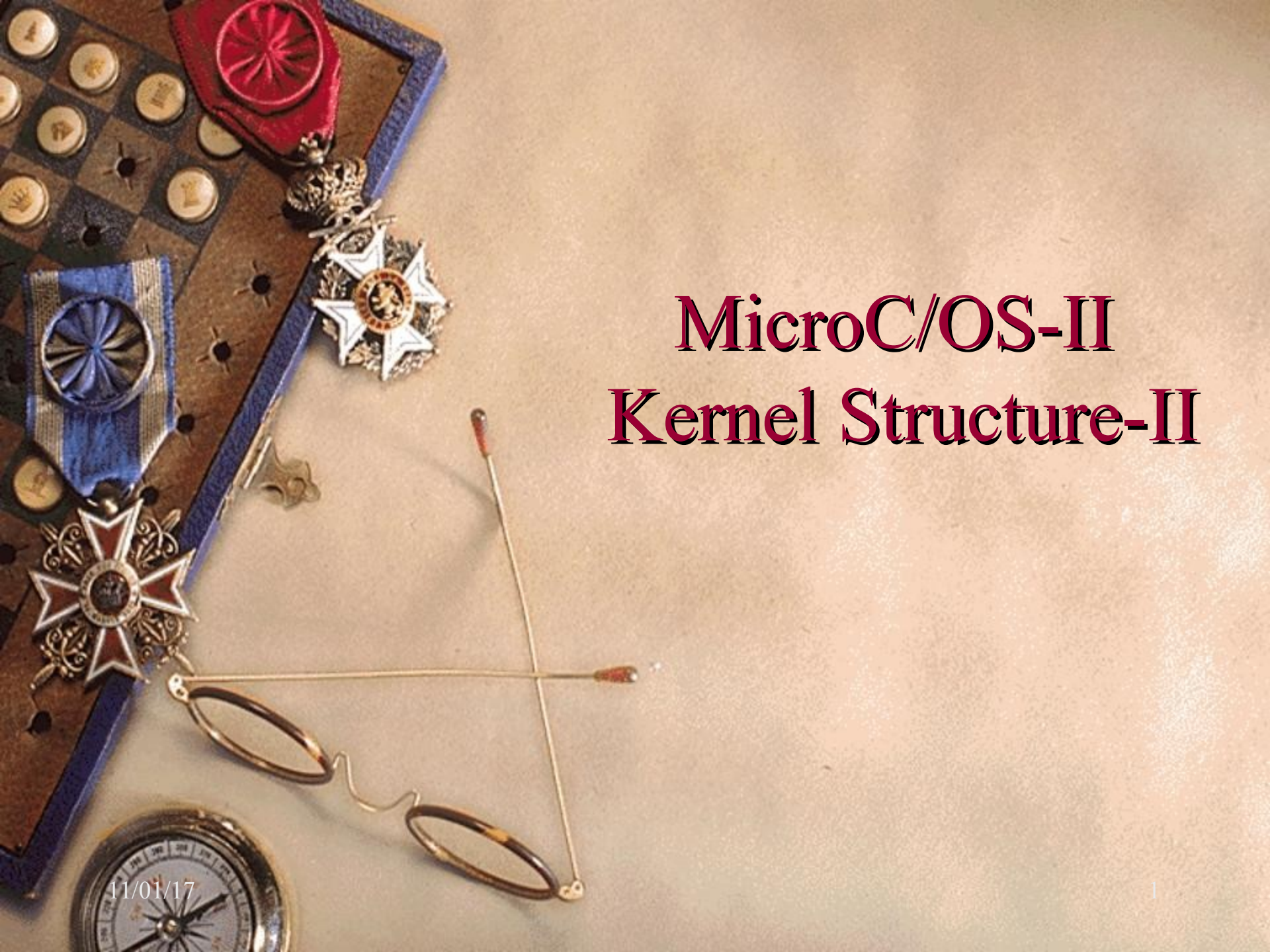


MicroC/OS-II
Kernel Structure-II

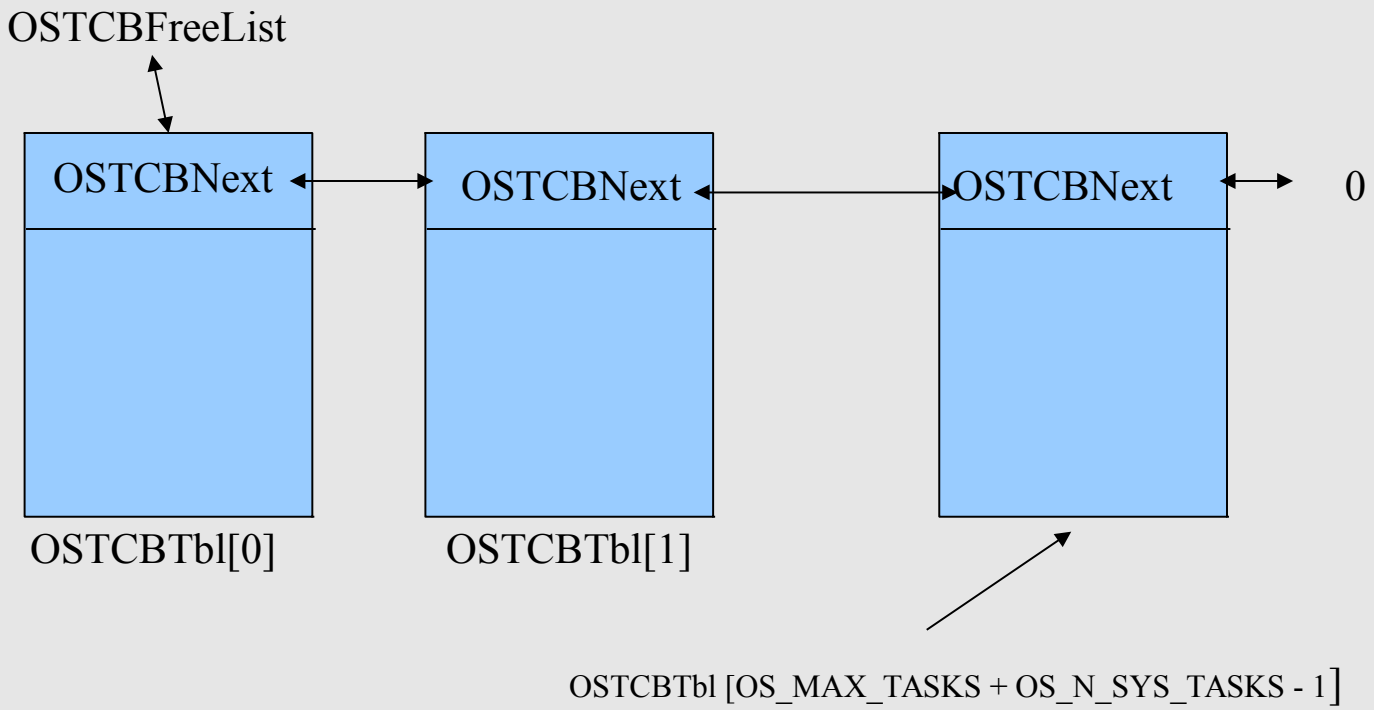
Task Control Blocks

- 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

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



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

#if 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                  */

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

#if (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           */

#if 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          OSTCBId;
#endif
```

*struct os_tcb *OSTCBNext;*
*struct os_tcb *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)

```
INT16U       OSTCBDly;
INT8U        OSTCBStat;
INT8U        OSTCBPrio;
INT8U        OSTCBX;
INT8U        OSTCBY;
INT8U        OSTCBBitX;
INT8U        OSTCBBitY;
```

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];
```

Ready List

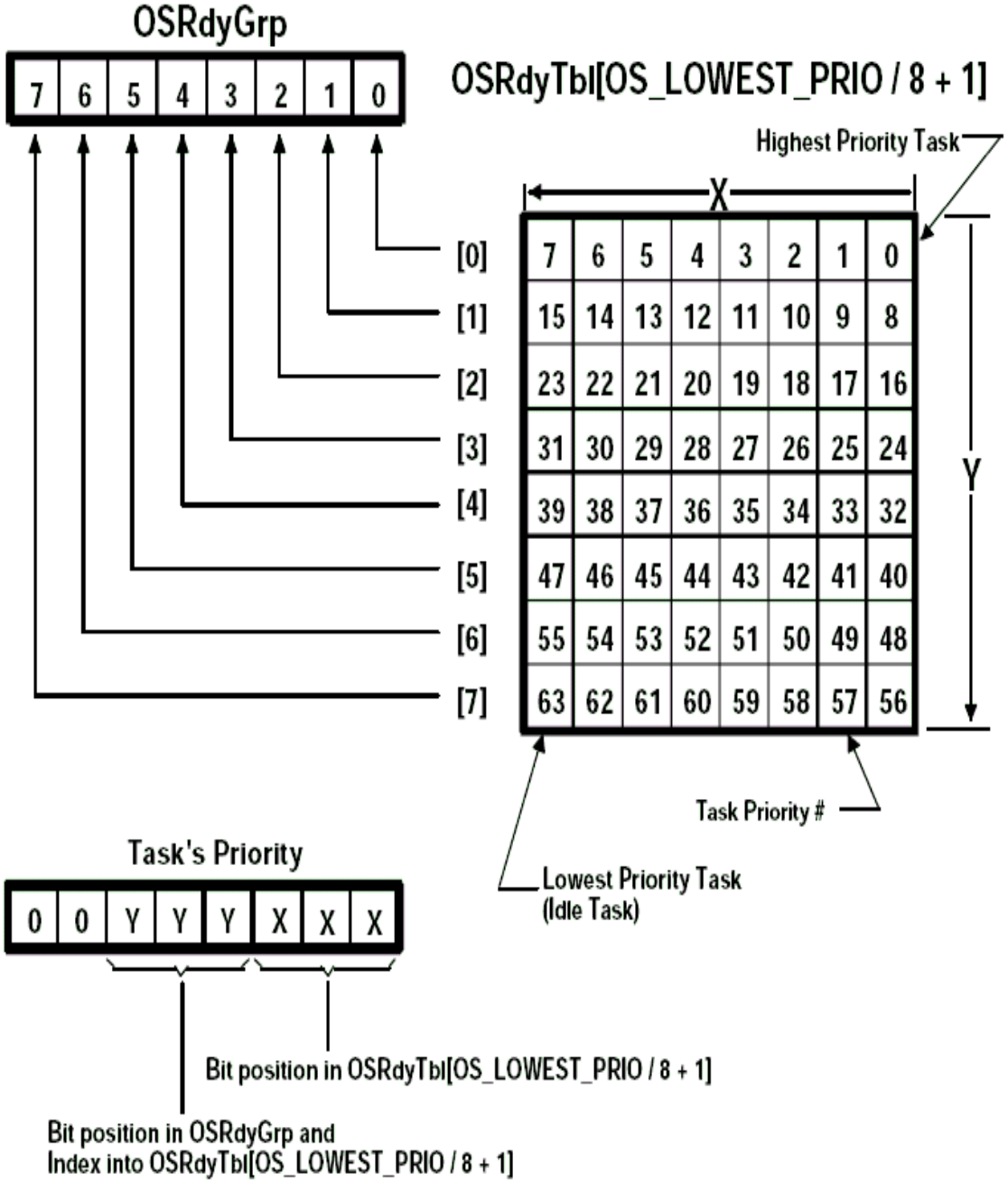
- A task owns a priority level between 0 and OS_LOWEST_PRIO.
- Each task that is ready to run is placed in Ready List.

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.

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.


OSRdyGrp
7 6 5 4 3 2 1 0
OSRdyTbl[OS_LOWEST_PRIO / 8 + 1]
Highest Priority Task
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
Task Priority #
Lowest Priority Task
(Idle Task)
Task's Priority
0 0 Y Y Y X X X
Bit position in OSRdyTbl[OS_LOWEST_PRIO / 8 + 1]
Bit position in OSRdyGrp and
Index into OSRdyTbl[OS_LOWEST_PRIO / 8 + 1]

Ready List

- `OSMapTbl[]`

  OSMapTbl[0] == 00000001b
  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    = OSUnMapTbl[OSRdyGrp];
x    = OSUnMapTbl[OSRdyTbl[y]];
prio = (y << 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

The End