Chapter 3
Kernel Structure - Part II

Ming-Hsin Ho

2008/07/22

Outline

- Locking and Unlocking the Scheduler
- Idle Task
- Statistics Task
- Interrupts Under MicroC/OS-II
- Clock Tick
- MicroC/OS-II Initialization
- Starting MicroC/OS-II
- Obtaining the Current MicroC/OS-II Version

Locking and Unlocking the Scheduler (1/3)

- The OSSchedLock() function is uesd to prevent task rescheduling until its counterpart OSSchedUnlock() is called.
- OSSchedLock() and OSSchedUnlock() must be used in pairs.

Locking and Unlocking the Scheduler (2/3)

```
1  void OSSchedLock (void)
2  {
3      if (OSRunning == TRUE) {
4          OS_ENTER_CRITICAL();
5          if (OSLockNesting < 255) {
6              OSLockNesting++;
7          }
8          OS_EXIT_CRITICAL();
9      }
10 }
```

Locking and Unlocking the Scheduler (3/3)

```
1  void OSSchedUnlock (void)
2  {
3      if (OSRunning == TRUE) {
4          OS_ENTER_CRITICAL();
5          if (OSLockNesting > 0) {
6              OSLockNesting--;
7              if ((OSLockNesting == 0) && (OSIntNesting == 0)) {
8                  OS_EXIT_CRITICAL();
9                  OS_Sched();
10             } else {
11                 OS_EXIT_CRITICAL();
12             }
13         } else {
14             OS_EXIT_CRITICAL();
15         }
16     }
17 }
```

Idle Task (1/2)

- MicroC/OS-II always creates a idle task.
- The idle task is executed when none of the other tasks are ready to run.
- The idle task is always set to the lowest priority.
  - OS_LOWEST_PRIO

Idle Task (2/2)

```
1  void OS_TaskIdle (void *pdata)
2  {
3          pdata = pdata;
4          for(; ;) {
5              OS_ENTER_CRITICAL();
6              OSIdleCtr++;
7              OS_EXIT_CRITICAL();
8              OSTaskIdleHook();
9          }
10 }
```

You can use OSTaskIdleHook() to STOP the CPU so that it can Enter low-power mode.

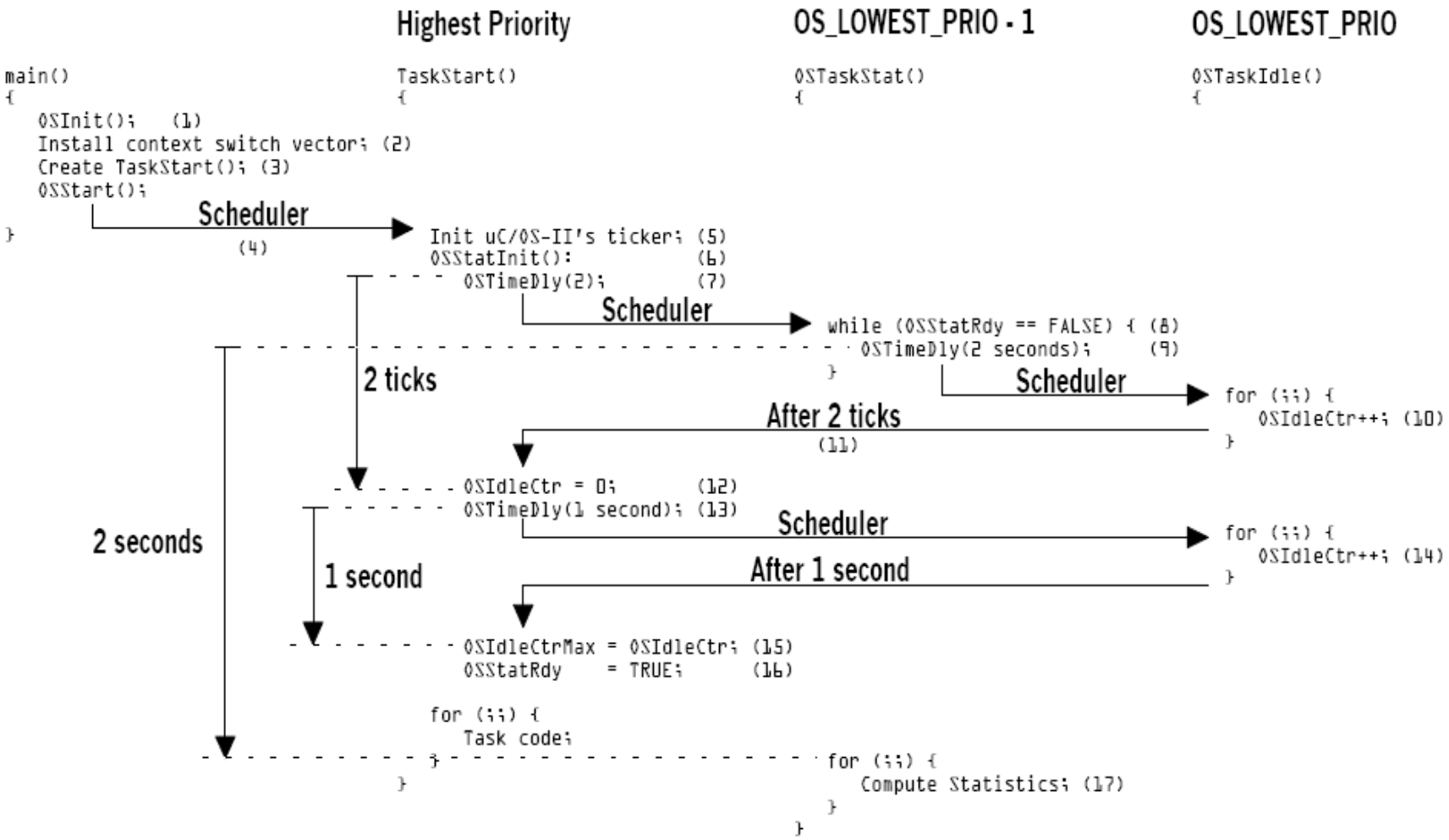
Statistics Task (1/7)

- It is created by MicroC/OS-II if you set the configuration constant OS_TASK_STAT_EN to 1.
- When enable, It executes every second and computes the percentage of CPU usage.

Statistics Task (2/7)

```
1  void main (void)
2  {
3        OSInit();
4        /* Install MicroC/OS-II's context switch vector */
5        /* Create your startup task (for sake of discussion, TaskStart()) */
6        OSStart();
7  }
8
9  void TaskStart (void)
10 {
11       /* Install and initialize MicroC/OS-II's ticker */
12       OSStatInit();
13       /*Create your application task(s) */
14       for (; ;) {
15           /* Code for TaskStart() goes here */
16       }
17 }
```

Statistics Task (3/7)



Statistics Task (4/7)

```
1  void OSStatInit (void)
2  {
3          OSTimeDly(2);
4          OS_ENTER_CRITICAL();
5          OSIdleCtr = 0L;
6          OS_EXIT_CRITICAL();
7          OSTimeDly(OS_TICKS_PER_SEC);
8          OS_ENTER_CRITICAL();
9          OSIdleCtrMax = OSIdleCtr;
10         OSStatRdy = TRUE;
11         OS_EXIT_CRITICAL();
12 }
```

Statistics Task (5/7)

```
1   void OS_TaskStat (void *pdata)
2   {
3           INT32U run;
4           INT32U max;
5           INT8S usage;
6           pdata = pdata;
7           while (OSStatRdy == FALSE) {
8               OSTimeDly(2 * OS_TICKS_PER_SEC);
9           }
10          max = OSIdleCtrMax / 100L;
11          for (; ;) {
12              OS_ENTER_CRITICAL();
13              OSIdleCtrRun = OSIdleCtr;
14               run = OSIdleCtr;
15              OSIdleCtr = 0L;
16              OS_EXIT_CRITICAL();
```

Statistics Task (6/7)

```
17          if (max > 0L) {
18              usage = (INT8S) (100L – run / max);
19              if (usage >= 0) {
20                  OSCPUUsage = usage;
21              } else {
22                  OSCPUUsage = 0;
23              }
24          } else {
25              OSCPUUsage = 0;
26              max = OSIdleCtrMax / 100L;
27          }
28          OSTaskStatHook();
29          OSTimeDly(OS_TICKS_PER_SEC);
30      }
31  }
```

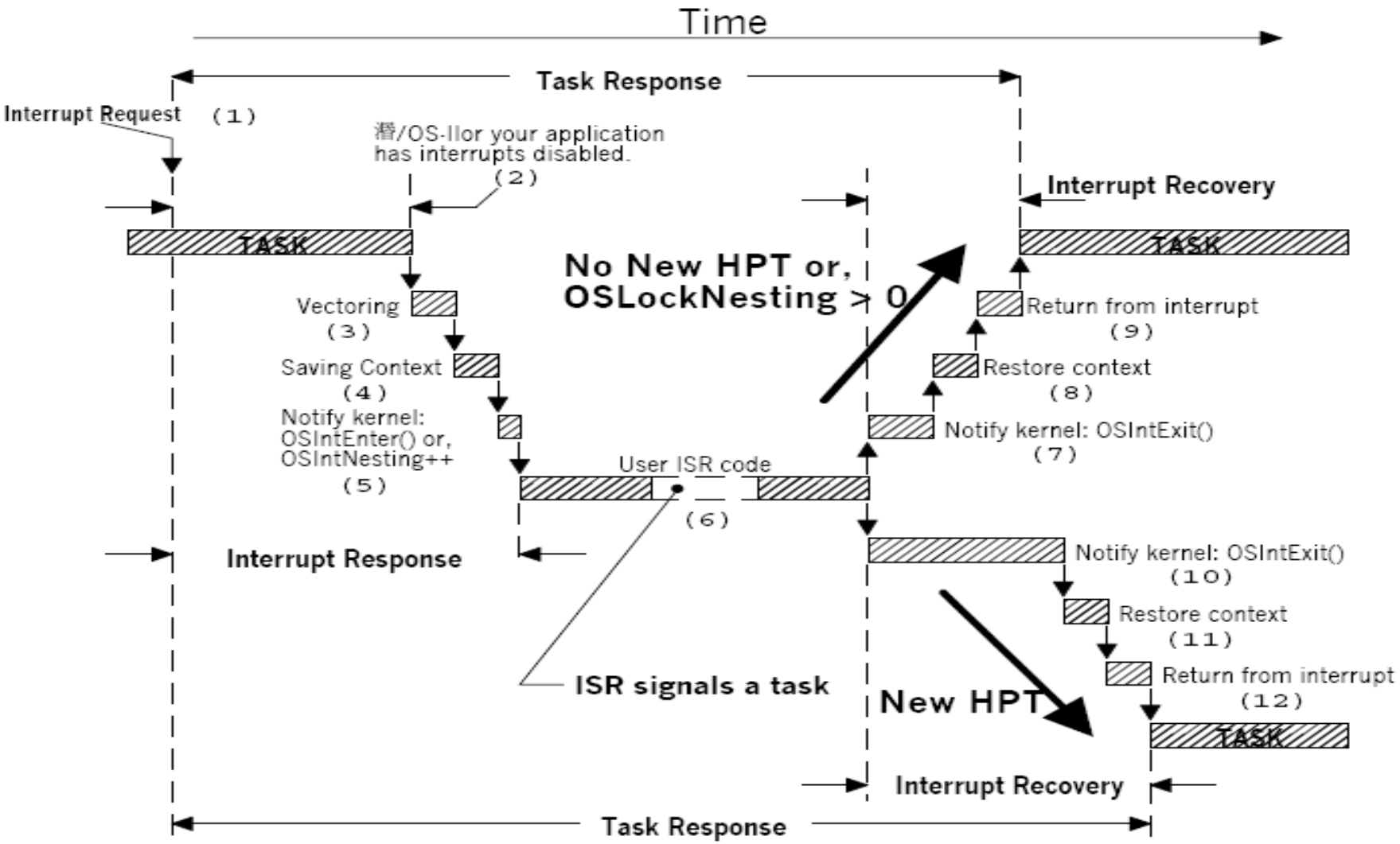
Statistics Task (7/7)

- OSCPUUsage(%) = 100 * (1 - OSIdleCtr / OSIdleCtrMax)
  - OSIdleCtr / OSIdleCtrMax = 0
- OSCPUUsage(%) = 100 - 100 * OSIdleCtr / OSIdleCtrMax
  - (100 * OSIdleCtr) limits the maximum value that OSIdleCtr can take
- OSCPUUsage(%) = 100 – OSIdleCtr / (OSIdleCtrMax / 100)

Interrupts Under MicroC/OS-II (1/4)

```
YourISR:
1       Save all CPU registers;
2       Call OSIntEnter() or, increment OSIntNesting directly
3       if (OSIntNesting == 1) {
4           OSTCBCur->OSTCBStkPtr = SP;
5       }
6       Clear interrupting device;
7       Re-enable interrupts (optional)
8       Execute user code to service ISR;
9       Call OSIntExit();
10      Restore all CPU registers;
11      Execute a return from interrupt instruction;
```

Interrupts Under MicroC/OS-II (2/4)



Interrupts Under MicroC/OS-II (3/4)

```
1  void OSIntEnter (void)
2  {
3          if (OSRunning == TRUE) {
4              if (OSIntNesting < 255) {
5                  OSIntNesting++;
6              }
7  }
```

Interrupts Under MicroC/OS-II (4/4)

```
1   void OSIntExit (void)
2   {
3           OS_ENTER_CRITICAL();
4           if (OSRunning == TRUE) {
5               if (OSIntNesting > 0) {
6                   OSIntNesting--;
7               }
8               if ((OSIntNesting == 0) && (OSLockNesting ==0)) {
9                   OSIntExitY = OSUnMapTbl[OSRdyGrp];
10                  OSPrioHighRdy = (INT8U) ((OSIntExitY << 3)
                                        + OSUnMapTbl[OSRdyTbl[OSIntExitY]]);
11                  if (OSPrioHighRdy != OSPrioCur) {
12                      OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
13                      OSCtxSwCtr++;
14                      OSIntCtxSw();
15                  }
16          }
17          OS_EXIT_CRITICAL();
18  }
```

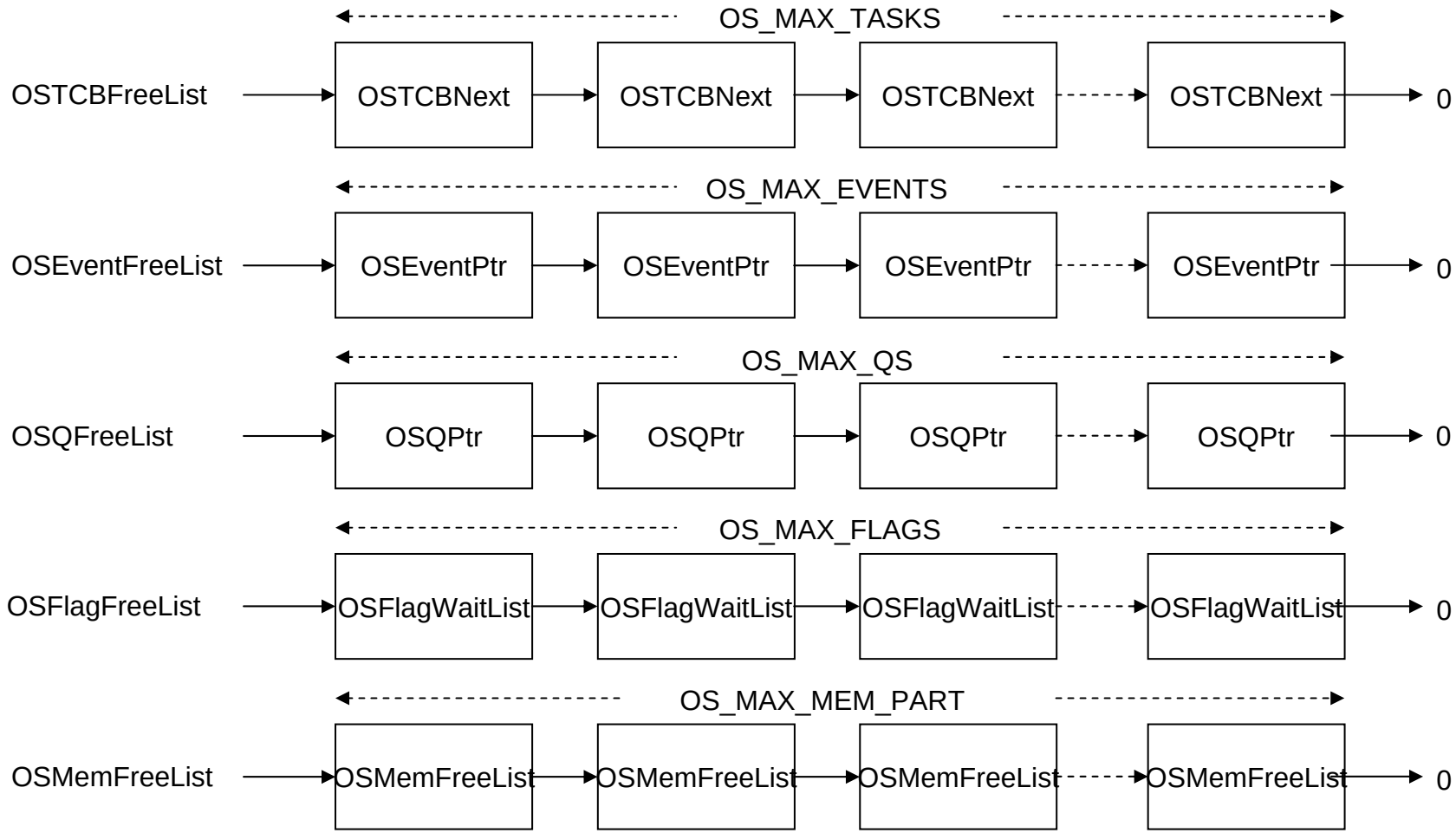
Clock Tick

- MicroC/OS-II requires that you provide a periodic time source to keep track of
  - time delays
  - timeouts
- You can obtain a tick source by
  - a hardware timer
  - an AC power line (50/60Hz) signal

MicroC/OS-II Initialization (1/2)

- It initializes all MicroC/OS-II variables and data structures.
- It creates the idle task. (OS_LOWEST_PRIO)
- It creates the statistic task. (OS_LOWEST_PRIO - 1)
  - OS_TASK_STAT_EN = 1
  - OS_TASK_CREATE_EXT_EN = 1

MicroC/OS-II Initialization (2/2)



Starting MicroC/OS-II (1/2)

```
1  void main (void)
2  {
3      OSInit();
4
5      Create at least 1 task using either OSTaskCreate() or OSTaskCreateExt()
6
7      OSSatrt();
8  }
```

Starting MicroC/OS-II (2/2)

```
1  void OSStart (void)
2  {
3      INT8U y;
4      INT8U x;
5      if (OSRunning == FALSE) {
6          y             = OSUnMapTbl[OSRdyGrp];
7          x             = OSUnMapTbl[OSRdyTbl[y]];
8          OSPrioHighRdy = (INT8U) ((y << 3) + x);
9          OSPrioCur     = OSPrioHighRdy;
10         OSTCBHighRdy  = OSTCBPrioTbl[OSPrioHighRdy];
11         OSTCBCur      = OSTCBHighRdy;
12
13         OSStartHighRdy();
14     }
15 }
```

Obtaining the Current MicroC/OS-II Version

```
1   INT16 OSVersion (void)
2   {
3           return (OS_VERSION);
4   }
```

Return the version number, multiplied by 100.

In other words, version 2.52 is returned as 252.

The End