4 Time Management

Contents

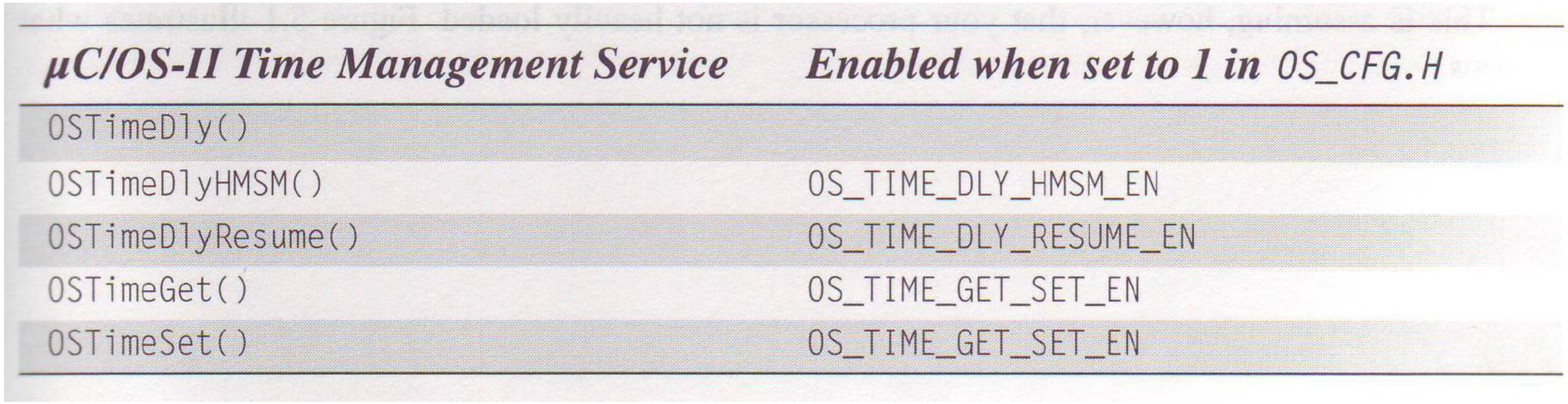
- Delaying a Task: `OSTimeDly()`
- Delaying a Task: `OSTimeDlyHMSM()`
- Resuming a Delayed Task:

  `OSTimeDlyResume()`
- System Time: `OSTimeGet()`, `OSTimeSet()`

Time Management Functions

- This chapter describes five services that deal with time issues (`OS_TIME.C`)
  - `OSTimeDly()`
  - `OSTimeDlyHMSM()`
  - `OSYimeDlyResume()`
  - `OSTimeGet()`
  - `OSTimeSet()`

Time Management Configuration Constants (`OS_CFG.H`)



µC/OS-II Time Management Service	*Enabled when set to 1 in* `OS_CFG.H`
`OSTimeDly()`	
`OSTimeDlyHMSM()`	`OS_TIME_DLY_HMSM_EN`
`OSTimeDlyResume()`	`OS_TIME_DLY_RESUME_EN`
`OSTimeGet()`	`OS_TIME_GET_SET_EN`
`OSTimeSet()`	`OS_TIME_GET_SET_EN`

Delaying a Task: OSTimeDly()

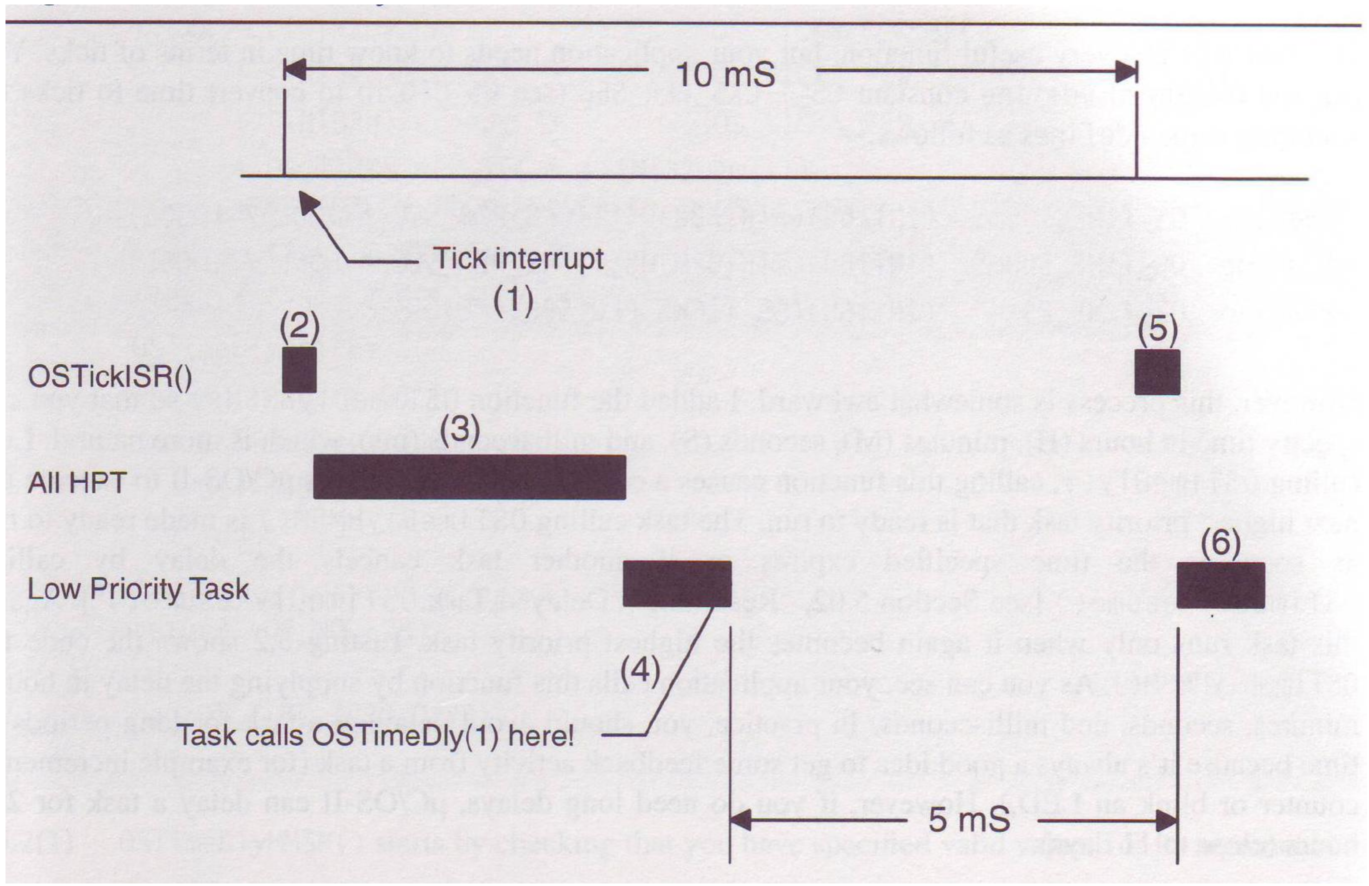
```
void  OSTimeDly (INT16U ticks)
{
#if OS_CRITICAL_METHOD == 3
  OS_CPU_SR  cpu_sr;
#endif

  if (ticks > 0) {
    OS_ENTER_CRITICAL();
    if ((OSRdyTbl[OSTCBCur->OSTCBY] &= ~OSTCBCur->OSTCBBitX) == 0) {
      OSRdyGrp &= ~OSTCBCur->OSTCBBitY;
    }
    OSTCBCur->OSTCBDly = ticks;
    OS_EXIT_CRITICAL();
    OS_Sched();
  }
}
```

maximum tick is 2^16

Removing a task from the ready list (see chapter 3)

ticks value is stored in TCB

Delay Resolution



If your application want to delay at least one tick, you must call OSTimeDly(2)

Delaying a Task: `OSTimeDlyHMSM()`

```
INT8U OSTimeDlyHMSM(INT8U hours, INT8U minutes, INT8U seconds, INT16U milli)
{
  INT32U ticks; INT16U loops;

  if (hours > 0 || minutes > 0 || seconds > 0 || milli > 0) {
    ...
    ticks = ((INT32U)hours * 3600L + (INT32U)minutes * 60L +
                 (INT32U)seconds) * OS_TICKS_PER_SEC +
      OS_TICKS_PER_SEC * ((INT32U)milli + 500L / OS_TICKS_PER_SEC) / 1000L;
    loops = (INT16U)(ticks / 65536L);
    ticks = ticks % 65536L;
    OSTimeDly((INT16U)ticks);
    while (loops > 0) {
      OSTimeDly(32768); OSTimeDly(32768); loops--;
    }
    return (OS_NO_ERR);
  }
  return (OS_TIME_ZERO_DLY);
}
```

convert time to tick number

convert tick number to loops and ticks (note 2^16=65536)

call OSTimeDly multiple times

Resuming a Delayed Task: OSTimeDlyResume()

```
INT8U  OSTimeDlyResume (INT8U prio)
{
   OS_TCB *ptcb;

   ...
   OS_ENTER_CRITICAL();
   ptcb = (OS_TCB *)OSTCBPrioTbl[prio];
   if (ptcb != (OS_TCB *)0) {
     if (ptcb->OSTCBDly != 0) {
       ptcb->OSTCBDly  = 0;
       if ((ptcb->OSTCBStat & OS_STAT_SUSPEND) == OS_STAT_RDY) {
         OSRdyGrp               |= ptcb->OSTCBBitY;
         OSRdyTbl[ptcb->OSTCBY] |= ptcb->OSTCBBitX;
         OS_EXIT_CRITICAL();
         OS_Sched();
       } else {
         OS_EXIT_CRITICAL();
       }
       return (OS_NO_ERR);
     } else {
       OS_EXIT_CRITICAL();
       return (OS_TIME_NOT_DLY);
     }
   }
   OS_EXIT_CRITICAL();
   return (OS_TASK_NOT_EXIST);
}
```

check if the task exists and the delay value is not 0

set TCB's delay value 0

if task is not suspended

make the task ready

OSTCBStat:

OS_STAT_RDY (=0x00): Ready to run
OS_STAT_SEM (=0x01): Pending on semaphore
OS_STAT_MBOX (=0x02): Pending on mailbox
OS_STAT_Q (=0x04): Pending on queue
OS_STAT_SUSPEND (=0x08): Task is suspended
OS_STAT_MUTEX (=0x10): Pending on mutual exclusion semaphore
OS_STAT_FLAG (=0x20): Pending on event flag group

System Time: `OSTimeGet()`

```
INT32U  OSTimeGet (void)
{
#if OS_CRITICAL_METHOD == 3
  OS_CPU_SR  cpu_sr;
#endif
  INT32U     ticks;

  OS_ENTER_CRITICAL();
  ticks = OSTime;
  OS_EXIT_CRITICAL();
  return (ticks);
}
```

the variable OSTime holds the current tick value (note: 32bit counter with 100 Hz rolls over every 497 days)

System Time: OSTimeSet()

```
void  OSTimeSet (INT32U ticks)
{
#if OS_CRITICAL_METHOD == 3
  OS_CPU_SR  cpu_sr;
#endif

  OS_ENTER_CRITICAL();
  OSTime = ticks;
  OS_EXIT_CRITICAL();
}
```

change the current tick value